ファインマン 計算機科学

A. ヘイ，R. アレン 編

原 康夫，中山 健，松田和典 訳

岩波書店

Feynman LECTURES ON COMPUTATION

RICHARD P. FEYNMAN
ANTHONY J. G. HEY
ROBIN W. ALLEN

ファインマン計算機科学

ファインマン計算機科学

A. ヘイ，R. アレン 編

原 康夫，中山 健，松田和典 訳

岩波書店

FEYNMAN LECTURES ON COMPUTATION

by Richard P. Feynman

Copyright © 1996 by Carl R. Feynman and Michelle Feynman

Foreword and Afterword copyright © 1996 by Anthony J.G. Hey and Robin W. Allen
First published in the United States by Perseus Books, a Member of the Perseus Group, Reading, Massachusetts 01867 U.S.A.
This Japanese edition published 1999 by Iwanami Shoten, Publishers, Tokyo, by arrangement with Perseus Books, Reading, Massachusetts.

編者の序

ファインマンが亡くなってから8年ほどになるので，私はこの『ファインマン計算機科学』の誕生のいきさつについて説明する必要があると感じている．1987年11月に私は長年ファインマンの秘書であったヘレン・タックから電話を受けた．ファインマンが私に計算機科学の講義ノートを出版するために編集してほしいと言っているという．その16年前に，私はカルテク*1の研究員(ポスドク)であったが，自分の研究がおろそかになるという理由でファインマンの「パートン」についての講義録の編集依頼を辞退していた．私はこのときの判断をしばしば後悔していたので，今回の依頼についてはすぐにやってみようという気になった．カルテクに最初に在籍していたとき，私は素粒子物理学者であったが，10年後の1981年にサバティカルでそこを訪れたときには，私は計算物理学の問題に興味を持つようになっていた．当時の私はファインマンがその何年も前に使った技法に似た変分法(後でわかったのであるが)を使って研究を楽しんでいた．そしてそのころカルテクで聴いたカーバー・ミードの「VLSIの将来」という講演に触発されて，並列計算や計算機科学の研究を始めることになった．

ファインマンは，マンハッタン計画でのプルトニウム爆弾のモデル計算以来，計算(computing)に長年興味を持っていた．『ご冗談でしょう ファインマンさん！』の中の「下から見たロスアラモス」で，ファインマンは爆縮時のエネルギー放出の計算を担当した「IBMグループ」をどのように監督したかを回想している．ディジタルコンピューターが出現する前であったが，ファインマンと彼のチームは爆弾の計算を並列化する方法を考え出した．

1981年にファインマンが，ジョン・ホップフィールド，カーバー・ミードといっしょに担当した「計算の物理学」と題する学際コースがカルテクの公式記録に載っている．このコースは2年間開講されたが，ジョン・ホップフィールドによると3人がともに同じ年に講義をしたことはなかったそうである．1年目はファインマンが病気で，2年目はミードが休暇をとっていた．1982-83年の授業で配布されたプリントには，計算の基礎入門，計算可能性と情報理論に続いて「物理的世界で生じる計算の限界と計算の“基本的”限界」と題された章が記されていて，講義の雰囲気がわかる．その年の講義はファインマ

*1 California Institute of Technologyの略.

ンとホップフィールドが担当し，マービン・ミンスキー，ジョン・コウクやチャールズ・ベネットのような専門家たちの招待講義が行なわれた．1983年の春に，ファインマンは，MITおよび彼の息子のカールとの関係で，野心的な新しい並列計算機の会社シンキング・マシンズのダニー・ヒリスのコンサルタントとして働いた．

1983年秋に，ファインマンは初めて一人で計算についてのコースを担当した．このコースは「計算機械の可能性と限界」としてカルテクの記録に載っている．この講義は1984, 85年と1985, 86年に録音されたのだが，この講義録はそのテープとファインマンのノートから復元されたものである．私はヘレン・タックへの返事に，1988年1月に「ハイパーキューブ会議」で講演するためにカルテクを訪問する予定であると書いた．これは並列計算の会議で，カルテクでのジェフリー・フォックスやチャック・サイツによる「Cosmic Cube」並列計算機の先駆的研究に端を発していた．私はその1月にファインマンと話をしたが，彼は自分の計算機科学の講義が日の目を見ることを熱望していた．私はこの計画を引き受けることを約束し，連絡を取り続けると約束してサウサンプトンに帰った．悲しいことに，ファインマンはこの会合の後，間もなく亡くなってしまい，彼が提案した講義録の内容についてさらに詳しく話し合う機会はなくなった．

ヘレン・タックはファインマンのノートとテープの両方のコピーを私に送り届けてきた．彼の講義録を出版物にふさわしい形にするのは膨大な作業量が必要であることがわかった．ホップフィールドやミードといっしょに担当した以前の講義と同じように，何人かの招待講義者が，プログラミング言語の「Scheme」から「Cosmic Cube」への物理学の応用に至る話題について1つあるいは複数の講義をしていた．また，私は以前に何人かがこの編集の仕事を試みていたことを発見した！ ファインマンの講義の基本的な核心はすぐに明らかになった．すなわち，コンピューター入門の部分，およびその後につづく論理ゲートの構造，数理論理学，構成部品の非信頼性，計算の熱力学，半導体技術の物理学等に起因するコンピューターの限界を探求した5つの部分である．6番目の部分では，ファインマンは量子力学によるコンピューターの限界について議論している．彼の量子力学的コンピューターの分析は1984年7月にアナハイムでの会合で発表され，つづいて1985年2月に雑誌 *Optics News* に論文が掲載された．これらの部分の後に，コンピューターの広範囲の「先端的応用」についての招待講義の部分が続いた．そのテーマはロボット工学，人工知能，ビジョン，並列アーキテクチャなど，年によって変わっている．

コースの題名に示されているように，ファインマンは講義でコンピューターの限界と可能性を探求しようと試みている．この講義は十年

あまり前に行なわれたものであるが，内容の多くは比較的「時代を超越したもの」であり，計算機科学のいくつかの標準的な話題についてファインマン流の見解を述べている．しかし，このコースは全体として独特なものであり，しかも正真正銘，学際的である．計算可能性のような題目の「ファインマン流の取り扱い」，チューリング機械(またはファインマンが言っているように，「チューリングさんの機械」)，シャノンの定理や情報理論に加えて，ファインマンは可逆計算，熱力学および量子計算についても議論している．コンピューターの重要な基本についてのこれほどまでの幅広い議論は疑いなく類がないものであり，計算全体についてのファインマン流の見方の「わき道」である．これは計算の全貌がこの講義で議論されているという意味ではない．多くの省略がある．2つの例を挙げれば，プログラミング言語とオペレーティングシステムである．それにもかかわらず，この講義はディジタルコンピューターの真に基本的な限界についてのわれわれの知識のまとめになっている．ファインマンは専門の計算機科学者ではなかったが，末梢的なことを探求するよりも本質的な事柄を強調し，非常に手早く多くの内容について論じている．それにもかかわらず，この課題への彼の迫り方はあくまでも現実的であり，これはチューリング機械の議論を通して計算可能性の理論を取り扱っていることで裏付けられている．チューリング機械のように一見とても単純なものから，どのようにしてこのように重大な結論にたどり着けるかの説明を，ファインマンが楽しんでいるのは明らかである．彼の学習と発見に対する哲学も，この講義に強く現れている．ファインマンは，「専門家」がどのようにやったのかを知るために本を読む前に自分自身で解決すること，いろいろ試して楽しむことの重要性をたえず強調している．この講義を読めばファインマンの研究方法について，他では得られない洞察が得られる．

私はところどころで編集者の特権を使ったが，それについてここで説明しておく．[RPF]という脚注がところどころにあるが，これは，ファインマンが講義で話した余談である．文章上，脚注とするのがふさわしいと思われるのでこうしてある．[編者]と書かれた脚注は，私と共編者のロビン・アレンのコメントである．なお，ファインマンの使っている表記法を現在の慣用表現に改めているところがいくつかある．たとえばMOSトランジスタの表し方である．

ファインマンは計算機科学を通常の方法で学ばなかった．彼の学び方の典型的な場合はこうである．同僚が彼の関心を引きつけそうなことを話すと，彼は一人になって，その問題を詳細に研究して解決する．このような独力による問題解決の過程で，いくたびかファインマンは課題に新しい光を当てることができた．量子コンピューター(quantum computation)についての彼の分析はまさにその例であるが，こ

の方法の他の研究者に対する欠点の例証でもある．量子コンピューターについての論文の参考文献の後にファインマンらしい脚注がある．それには「T・トフォリの参考文献が役立ったことに感謝の意を表す」と書かれている．すばらしい洞察力と明晰な思考によって，ファインマンはしばしば現実にいくつかの進歩をもたらせたばかりでなく，問題全体の基礎を明らかにすることもできた．その結果，ファインマンによる量子コンピューターの論文は，この分野に重要な貢献をした他の少数の人を除いて，広く引用されている．これについては，ファインマンが最初にこの問題をチャールズ・ベネットから聞いたためベネットは頻繁に引用されているが，ロルフ・ランダウアーやポール・ベニオフのような他の先駆者たちは省略されている．私はファインマンが他人の功績を横取りするつもりがないことを確信しているので，勝手ながらこの歴史的記録文書の何カ所かに修正を加え，読者がこの問題のより完全な歴史を参照できるように脚注を付した．ファインマンは課題の歴史には関心がなく，実際に解くべき問題にのみ関心があったというのがありのままの真実である！

私はこの他にも編集者の特権を行使して，1980 年代なかば以降には時代遅れになったり，無用になった話題に関するいくつかの講義を除いた．しかし，このコースのより正確な印象を伝えるため，カルテクでこのコースに実際に参加した「専門家」が執筆した「先端的話題」を集録した姉妹編が出る予定である．この姉妹編は最近 10 年間の進歩に充てられ，ファインマンのコンピューター探求の記念碑としてふさわしいものになるだろう．

この計画が首尾よく完成したのを機会に，多くの人たちに感謝したい．とりわけ，無秩序だったノートを私のために整理してくれたサンディー・フレイとエリック・モルネスには感謝しなければならない．ジェフリー・フォックスにはこのコースを履修した学生を追跡してくれたことに，ロッド・ファン・メーターとタカコ・マトバには自分たちのノートのコピーを送ってくれたことに深謝したい．また，当初，この仕事を引き受けるように励ましてくれたことに対して，ゲリー・サスマンへの感謝とともに，故ヤン・バン・ド・スニープシュートへの謝意を特に記したい．ファインマンが一人でこのコースを担当することを決めたとき，ゲリーは休暇をとって MIT からカルテクに来ており，ファインマンがコースの計画を立てるのを手伝った．

この講義録の最終稿からは，(私の)理解の誤りをすべて確実に取り除こうと努力した．この作業では多くの人達の助けを借りた．ロルフ・ランダウアーは可逆計算と熱力学に関する第 5 章を親切に読んで改善するとともに，この問題の歴史を忍耐強く教えてくれた．スティーブ・ファーバーは ARM RISC プロセッサーの設計者であり，現在

はマンチェスター大学の教授になっているが，私には直接の知識がほとんどないVLSIに関する第7章を読んで詳細にコメントをしてもらった．サウサンプトン大学の何人かの同僚たちは，本文をまとめるにあたって大いに助けてくれた．第1章と第2章はアドリアン・ピッカリングとエド・ザルスカ，第3章はアンディ・グラベル，第4章はラホス・ハンゾ，第5章はクリス・アンソニー，そして第7章はピーター・アッシュバーン，ジョン・ハメル，グレッグ・パーカーおよびエド・ザルスカである．サウサンプトン大学のデビッド・バロン，ニック・バロンおよびマイク・クインとMITのトム・ナイトには原稿全体を読み通してもらったが，彼らのコメントのおかげで多くの誤りや不明瞭な箇所が取り除かれた．言うまでもなく，残っている誤りや混乱しているところはすべて私に責任がある！　カーディフ大学のボブ・チャーチハウスにはベーコン学派の暗号に関する情報について，サウサンプトン大学のボブ・ネスビットには地質学者ウィリアム・スミスに関して私を啓蒙してくれたことについて，そしてバス大学のジェイムズ・デブンポートには積分のアルゴリズムによる解法に関係する参考文献での援助について感謝しなければならない．また，1985年の*Optics News*に掲載された古典的論文である「量子力学的コンピューター」と題するファインマンの論文を，少し修正した形でこの講義録の第6章に再掲することを許可してくれたアメリカ光学会に対して感謝する．

ファインマンが亡くなった後，彼の妻のグウェネスとファインマン家の友人のダッドレー・ライトには大いに援助してもらった．その1つが講義テープを文書におこす作業への資金援助である．ファインマン文書の所有権についての長い法律論争に決着がついた後で，私がこの計画を再出発させるのを支援してくれた共編者のアレンにも感謝しなければならない．彼がいなければ，この計画は決して日の目を見なかっただろう．また，法的な膠着状態にあった長年の間，常にこの計画を支え，あらゆる援助を提供してくれたミシェル・ファインマンやカール・ファインマンと彼の妻ポーラに感謝する．この出版計画の初期に多大な誠意を示してくれたAddison-Wesley社の学術書プログラムの当時の責任者のアレン・ワイルドにお礼の言葉を述べたい．後期には，Addison-Wesley社の学術書部のジェフ・ロビンスとヒーザー・ミノーは不可避だった原稿の遅れと，見たところ重要に思われない細部へのイライラさせるほどの私の固執を立派に忍耐強く見守ってくれた．最後に，ヘレン・タックには彼女の私への信頼と，私が仕事を完成するという彼女の確信(私は信じていないときもあった！)にたいして謝意を記さねばならない．

1996年5月

トニー・ヘイ
英国サウサンプトン大学電子・計算機科学部

ファインマン序

私が今から約30年前に『ファインマン物理学』を著したとき，それらが物理学へ進もうとする学生の助けになると考えていた．それと同時に，私は数百年間の価値ある科学をわずか3巻の中に詰め込むことの難しさを痛感した．この『ファインマン計算機科学』では，それはいくらかやさしいが，ほんの少しだけである．まず，この講義は計算機科学の学生だけを対象にしたものではないので，私を試験のためのシラバスという束縛から解放してくれ，おもしろいという理由だけで講義の題材の分野を選ぶことを可能にしてくれる．つぎに，計算機科学は物理学ほど古くからあるわけではなく，200年の遅れがある．しかし，この事実は，物理学者の皿の上に比べて，計算機科学者の皿の上にははるかに少ない料理しか載っていないことを意味しない．計算機科学は新しいかもしれないが，はるかにもっと強烈な進歩があった！　そのために，われわれが取り上げることはたくさんある．

計算機科学は，実際には科学ではないという点でも物理学とは異なる．計算機科学は自然を対象として研究しない．数学的推論をかなり長々と使うが，数学でもない．計算機科学はむしろ工学のようなものである．スミスが現れる前の地質学*1のように，単に抽象概念を取り扱うだけでなく，むしろあることを行うために，あるものを手に入れていくという学問である．現在，計算機科学では「鉱山の中に入っていく」ことも必要である．──その後で一般化できる．まず細部を調べることは害にならない．

しかし，計算機科学はすべて実用的で，現実的な橋の建設のようなものだと言っているのではない．それはまったく違う．計算機科学は種々の深遠な問題に関わっている．われわれが理解していると思っていた言語の本質を，計算機科学は解明した．文法についての旧式の考え方では言語の基本的な要素のすべてをとらえることができなかったので，機械翻訳の初期の試みは失敗した．当然のこととして，この事

*1　ウイリアム・スミスは現代地質学の父であった．彼は運河や鉱山の技術者としての仕事の中で，岩石が系統的な層になっていることを観察し，地層の年代を決定する手段として化石が重要であることを認めた．このようにして，彼は古い層の上につぎつぎに岩石が堆積していくという「地層累重の法則」を明確に述べた．スミスのこの偉大な業績の前までの地質学は，経験科学よりも思弁哲学に近いものであった．［編者］

実は計算可能性について，すなわち，われわれの身のまわりの世界について何を知ることができて，何を知ることができないかについて疑問に思うことを勧めている．計算機科学の研究者たちは，人間が単なる機械であるかどうか，人間の頭脳がいつの日かコピーできる単に強力なコンピューターであるかどうかについて話すのに多くの時間を費やしている．そして「人工知能」の分野──私は「先端的応用」という言葉の方が好きだ──は，「人間の」知能や心の本質について多くのことを教えてくれるかもしれない．もちろん，脳の活動を研究すると役に立つ考えが得られるかもしれないが，自動車はチーターのような足をもっていないことや，飛行機の翼は羽ばたかないことを思い出さねばならない！　われわれは役立つ技術を生み出すために，生物の神経の詳細を研究する必要はない．しかし，間違った理論でさえ，機械を設計するのに助けになることがあるかもしれない．とにかく，計算機科学には単に技術的な興味以上のものがあることがわかるだろう．

この講義では，現在，機械を使って何ができて，何ができないかについて，またそれがなぜなのかについて語った．私は博士論文を書こうとするすべての学生に薦めるべき気分，「研究を開始するときに，賢かったが無知だった昔の自分に対して自分の考えを説明している様子を想像せよ！」，によってこの講義を行おうとした．いくつかの基礎概念を簡潔に紹介した後，つづく5つの章でコンピューターの限界──論理ゲートから量子力学まで──を探る，というのが本書のごく簡潔なあらましである．第2部は，私が先端的応用と呼んできた，ビジョン，ロボット工学，エキスパートシステム，チェス機械など*2についての専門家による講義で構成されている．

*2　この姉妹編は，現在，準備中である．この第2巻はファインマンのコースで招待講義した専門家による「先端的応用」に関する，現在の技術の状況を反映した論文をできる限り多く含むものになるだろう．〔編者〕

目　次

第1章
コンピューター入門

コンピューターは多くのことができる．瞬く間に何百万の数を足し算することができる．チェスの名人をうち負かすことができる．兵器を目標に誘導できる．君のために，ギターをつま弾く尼僧と煙草を吸わない物理学教授とに挟まれた飛行機の席を予約することもできる．ボンゴを叩けるものさえある．コンピューターは実にバラエティに富んでいる！　したがって，もしコンピューターについて語ろうとするのであれば，どのようなものをどのように調べようとするのかを，今ここで決めておいた方がよい．

われわれは個々の機械を調べることに多くの時間を費やすつもりはない．その理由は，いったんコンピューターの中身に立ち入ってみると，人間の場合と同様，多かれ少なかれ中身は類似しているからである．コンピューターには音楽を作り出すものもあれば画像を表示するものもあるし，またキーボードから動かすものもあれば，自動車の車輪の回転のトルクで動くものもある．というように，機能や入出力の種類が異なるものがあるが，その心臓部はよく似ている．そこでわれわれはコンピューターの内部機構について詳しく考えることにする．特定の入出力(I/O)構造や，機械に情報を入出力する方法については何も想定しないことにする．われわれの関心は，どのように入力があっても，それはディジタルの形をしていることと，何が出力されるにしても内部機構はディジタルで出力するということだけである．ディジタルとは2進数，すなわち1と0を意味する．

コンピューターの内部はどのようになっているのだろうか？　おおざっぱに言うと，単純な基本的素子から組み立てられている．この素子は特別なものではなくて，たとえば制御弁やそろばんの玉のようなものであってもよい．この基本的素子として何を用いるかには多くの選択の可能性がある．重要なのは，われわれが望むどのようなものを組み立てるのも，これらだけでできるということだけである．これらの素子はどのように組み合わされるのだろうか？　ここでも多くの選択の可能性があり，適切な構造はおそらく速度，エネルギー消費，美学などの考察によって決められる．このように眺めると，コンピューターの多様性は家の多様性に少しばかり似ている．高級住宅地のコンドミニアム(分譲マンション)はスラム街の車庫とはまったく違うよう

に見えるが，しかし両者は同じもの，たとえばブロック，モルタル，材木，労働で作られており，コンドミニアムはただそれらを多く使い，所有者の要望に応じてそれらの配置が異っているだけである．本質的には両者はとてもよく似ている．

少し抽象的になるが，「もっとも**多く**のことをするには，**どの**種類の素子を**どのように**接続したらよいのだろうか」という質問をしよう．これは奥の深い問題であるが，この場合にも「ある程度までは無関係だ」というのが答である．少しばかりのことができるコンピューターがあれば，厳密に言うと基本処理の「十分な一揃い(1組)」を持っているコンピューターがあれば，原理的には他のコンピューターができることは何でもできる．この事実は，厳密ではないが「万能性」という大原理の基礎になっている．「エーッ?!　私のポケット電卓はクレイ社のスーパーコンピューター群がやっているような木星の大赤斑をシミュレーションなんてできない！」と君たちは叫ぶだろう．いや，できるのである．ポケット電卓の配線をし直す必要があるだろうし，電卓のメモリーを増強する必要があるだろう．それにその電卓はひどく遅いだろうが，もし十分長い時間をかければクレイ社のスーパーコンピューター群が行なうことは何でも再現できるのである．一般に，2つのコンピューター **A, B** があって，**A** についてのすべてがわかっているとしよう．すなわちコンピューター **A** の働き方，その「状態遷移則」などである．まず，機械 **B** は **A** の状態を表すことができるとしよう．すると，われわれは **B** を使って **A** の遷移を順に記述していくことによって，**A** の動作をシミュレーションできる．言い換えると，**B** は **A** を真似るのである．もし **B** が非常に原始的で **A** が非常に精巧な場合にはこの作業にはきわめて長い時間がかかるが，**B** は，**A** のできることなら何でもいつかはできる．チューリング機械として知られている **B** のようなコンピューターを設計することによって，これを後で証明しよう．

万能性を別の見方で見てみよう．言語が良い例である．「あること，たとえば，ガソリン駆動の4輪車を表現するのに**最適な**言語は何か？」という質問を君たちにしよう．もちろん，ほとんどの言語には，少なくとも西洋では，これに対する簡単な言葉がある．アメリカでは「automobile」，イギリスでは「car」，フランスでは「voiture」である．しかし，「automobile」に対応する言葉を持つまでに進化していない言語があるだろう．そのような言語を話す人は，その言語の基本的な単語を使って見たままのことを表す．おそらく長くて複雑な表現を創造しなければならないだろう．しかしこれらのどの表し方も，他のどの表現より本質的に「優れている」ということはない．それぞれの表現はそれぞれその役割を果し，効率が違うだけである．単語のレベルで民主主義を導入する必要はない．アルファベット*[1]の階層にま

で降りてみよう．そしてたとえば，次のような質問をしよう．「英語にとって最良のアルファベットは何だろうか？」すなわち「なぜ普通の26文字を使いつづけるのだろうか？」アルファベット26文字で表現できるものはすべて，3つの記号「・」(トン)，「―」(ツー)，「　」(空白)によるモールス符号で表現できるし，あるいは2つの記号を使って5桁の2進数でAからZまでを表すベーコン学派の暗号を使っても表現できる．そのため，ここでも要素の基本的な一揃いの選択については多くの自由度があり，この選択が実際に影響するのは言語の効率，したがって本の厚さだけであることがわかる．つまり「最適な」言語やアルファベットは存在せず，それぞれは論理的に万能であり，それぞれは他のどれが表現するものも表現できる．計算での普遍性とは，基本処理の「十分な」一揃いを使って実行できる複雑な作業の範囲は，一揃いの基本処理の特定の詳細な構造には依存しないということである．

今日のコンピューターが複雑な作業を実行するためには，その作業を一連の簡単な基本処理によってどのように行なうかを示す正確で完全な記述，すなわち「ソフトウェア」が必要であり，これらの処理を明記された順序で実行していく機械，すなわち「ハードウェア」が必要である．この指示は正確で明確なものでなければならない．日常生活でわれわれは，言おうとすることを互いに**正確に**告げているわけではない．「正確に」告げなくても，前後関係やボディーランゲージ，相手をよく知っていることなどによって「ギャップを埋めて」，言われていることの曖昧さを解消できる．しかし，コンピューターは今のところ言われていることを人間のようには「理解」できない．行なうべきことをうんざりするほど詳しく正確にコンピューターに告げる必要がある．おそらくいつの日か，われわれはおおよその作業を記述するだけでうまく動く機械を持つようになるだろうが，そのときまではコンピューターが行なう作業をどのように告げるかについては非常に形式ばっていなければならない．

ある一揃いの基礎的な命令から複雑な命令を組み立てる方法について調べよう．たとえば命令の一揃いBが非常に単純であれば，複雑な処理をさせるにはとんでもなく多くの記述を必要とし，結果として「プログラム」は非常に長くて複雑になるのは明らかだろう．たとえば，コンピューターにいろいろな数値計算を実行させたいが，Bには掛け算が含まれていないとする．もし，この機械に3掛ける35を実行するように告げると，それは「何だって？」と言う．しかしBに足し算があれば，掛け算は足し算を何回も繰り返せばよいこと，すなわちこの場合は35に35を2回加えればよいことは考えればわかるだろう．しかし，掛け算をするようにBの基本命令を組み合わせた大

*1 「アルファベット」とは，ある言語で使用する文字の種類を表す．〔訳者〕

きな命令の塊によって**定義**された「掛け算」という命令をBに追加すれば，Bで書かれるプログラムの記述は明らかにすっきりする．2つの数を掛け合わせたいときには，「コンピューター，3掛ける35を実行せよ」と言うと，今度は「掛ける」という言葉を認識する．それは多くの足し算にすぎず，それを実行する．この機械は合成された命令を基本成分に分解し，われわれが低レベルの概念の泥沼にはまりこむことから救ってくれる．複雑な処理は，このように階層的に組み立てられている．これにとても似たことは日常生活でも起こっている．たとえば，われわれはさまざまな概念やそれらの結びつきを1つの言葉で置き換えている．われわれはこれらの概念や結びつきを引用するとき，1つの言葉だけですませ，元の多くの低レベルの概念にもどって作業することが避けられる．コンピューターはとても複雑な代物なので，このような概念の単純化が普通は必要で，細部の中で完全に迷ってしまうことを避けるためには良い設計が不可欠である．

われわれは基本的手続きの一揃いを構成することから始めてみよう．2つの数の足し算や2つの数をメモリー上のある場所から別の場所へ転送するという操作をどのように実行するのかを見てみる．その後で複雑さの階層構造の次の階層へ登り，これらの基本的な命令を使って掛け算などの演算を作ろう．この複雑さの階層構造をずっと上に登って行くつもりはない．もし，どこまで登って行くことができるのかを見たいのであれば，デニングとブラウンによるオペレーティングシステムについての論文(*Scientific American,* September 1984, 96-104)が13の階層を同定している．これは抵抗，ゲート，バスなどの電子回路素子の階層1からユーザーのプログラム環境を手際よく操作するオペレーティングシステム・シェルの階層13までの範囲にわたるものである．階層1での1と0の基本的な転送命令は，命令の階層的な合成によって階層13に達する頃には，シミュレーションで飛行機を着陸させたり40桁の数値が素数であるかどうかを調べる命令に変容している．われわれはこの階層構造のかなり下の方，しかしそこから上下に行けるような階層から始めよう．

われわれは，議論をいわゆる「ノイマン型アーキテクチャ」のコンピューターに限定する．建築を意味する「アーキテクチャ」という言葉を聞いて嫌にならないでほしい．アーキテクチャはれんがや柱ではなく，電子部品の接続方法を意味するもったいぶった言葉である．フォン・ノイマンは，量子力学の基礎へ重要な貢献をしたばかりでなく，現代的なコンピューターの基本原理について最初に明確に着手した有名な数学者である*2．われわれは同じ問題を解くときのいくつかのコ

*2　実際には現在では「非ノイマン型」機械の設計に多くの関心が持たれている．これらについてはこの講義に招待された「専門家」によって姉妹編のなかで議論されることになるだろう．[編者]

ンピューターの動作を後で調べるつもりだが，そのときは並列的にではなく逐次的に作動するコンピューター，すなわち問題の各部分を同時にではなく順番に解くコンピューターに限定する．「並列処理」を省略したことによって失うのは速度だけで，本質的ではない．

われわれは以前に，コンピューター科学は本当の科学ではないと言った．いま，この「コンピューター」という言葉とも縁を切らねばならない．「コンピューター」という言葉はわれわれに加算，減算，乗算などの算術計算を想像させるので，これがコンピューターが行なうことのすべてだと思いがちである．事実，普通のコンピューターには基本的な計算を行なう場所があるが，残りの部分がコンピューターの主要な仕事をするためのものである．それは紙切れをあちこち並べ替える仕事である．ただしこの場合，紙切れというのはディジタルの電気信号のことである．コンピューターはいろいろな意味で，事務室で威勢よくファイルキャビネットまで往復し，ファイルを出し入れし，多くの紙切れに記入し，メモを互いに受け渡すなどの作業を行なう，ファイル事務員の繁雑な作業を思い出させる．われわれはファイル事務員の作業にいくらか詳しく立ち入ることにするが，君たちの中で気の短い人はそれは細部に立ち入りすぎると思うかもしれない．しかし，それはコンピューターが行なうことの本質を伝えるのにぴったりなモデルなので，時間を費やす価値がある．

1-1　ファイル事務員モデル

多くのセールスマンを雇っている大会社があるとしよう．これらのセールスマンに関する膨大な情報が大きなファイルシステムに保存されていて，1人の事務員によってすべてが管理されている．この事務員がこのファイルシステムからどのようにして情報を得るかを知っていると考えて話を始めよう．データはカードに保存されていて，それぞれのカード(「売り上げ」カード)にはセールスマンの名前，担当地域，販売品目と販売数，給料，等々が書かれている．

セールスマン：＿＿＿＿＿＿＿＿
売り上げ：＿＿＿＿＿＿＿＿
給　料：＿＿＿＿＿＿＿＿
担当地域：＿＿＿＿＿＿＿＿
・
・
・

ここで，「カリフォルニアの全売り上げはいくらか？」という質問に対する答を求めていると仮定しよう．これはかなり退屈で単純な質問であるが，それが私がこの質問を選んだ理由である．難しい問題を理解するにはまず単純な問題から始めねばならない．それではファイ

ル事務員はどのようにしてカリフォルニアの全売り上げを知るのだろうか？　1つの方法がある：

カードを取り出す
もしそこに書かれている「担当地域」が**カリフォルニア**であれば，「売り上げ」欄に書かれた数値を「全売り上げ」と名付けられたこれまでの累計に加算する
カードをもどす
次のカードを取り出して同じことを繰り返す

明らかにすべてのカードについて初めから終わりまでこの作業をつづけなければならない．さて不幸にも特別に頭の悪い事務員を雇ってしまったとしよう．彼は読むことはできるが，上記の命令は彼には難しすぎると仮定しよう．たとえば，彼は累計をつづける方法がわからないとしよう．われわれは彼をもう少し援助する必要がある．事務員が使える「全売り上げ」カードを作ろう．彼は累計を保持するために次のような方法でこのカードを使うだろう：

次の「売り上げ」カードを取り出す
もし**カリフォルニア**であれば，
　「全売り上げ」カードを取り出す
　売り上げの数字を「全売り上げ」カードに書かれた数字に加える
　「全売り上げ」カードをもどす
「売り上げ」カードをもどす
次の「売り上げ」カードを取り出して同じことを繰り返す

これは幼稚なコンピューターがこの加算問題を解く方法の機械操作的な描写である．明らかに，データはカード上に保存されるわけではなく，またコンピューターが「カードを取り出す」必要はない．レジスターに保存された情報を読むのである．また，何かを物理的に置きにいくことをせず，レジスターから「カード」に書くことができる．

さて，われわれの事務員を能力いっぱいに働かせよう！　それぞれのセールスマンは会社から基本給を受け取るだけでなく売り上げから少しの歩合をもらうとしよう．販売成績に歩合の比率を掛ければ歩合給がいくらであるかがわかる．われわれは事務員に歩合給を計算してほしいと思っている．さてわれわれの事務員は低賃金で仕事は速い．しかし，不幸なことに彼は頭が悪く掛け算ができない*3．もし彼に「5掛ける7は？」と告げると，彼は「え，何？」と言う．そこで彼

*3　余談だが，われわれの頭の悪いファイル事務員はこの例では男性になっているが，男女差別を意味するものでは決してない！［RPF］

に掛け算を教えなければならない．これを行なうために，彼は非常に素早くカードを取ることができるという事実をうまく利用しよう．

2進数で作業しよう．君たちがおそらく知っているように，2進数の演算規則は10進数の規則よりやさしい．2進数の掛け算の表はとても小さく，カード1枚にたやすく書ける．われわれの事務員でもこの規則を覚えられると仮定しよう．次の例で明らかになるように，彼が覚えねばならないものは「桁移動(シフト)」と「桁上げ(繰り上がり)」だけである．

```
10進数では：     22×5 = 110
2進数では：       10110          10進数では：  22
                    101                          5
                 -------
                  10110
                 10110   (2回桁移動)           ___
                 -------
                1101110                        110
```

そこで，われわれの事務員が桁移動と繰り上げができれば，実質的に彼は掛け算ができることになる．彼は掛け算を泥臭く行なうが，非常に素早く行なう．それがここの要点である．コンピューターの中は，とても頭が悪いけれどもものすごい速さでこれを実行する．1秒間に単純な操作を何百万回も実行できるので，作業は非常に早いが頭の悪いファイル事務員のようなものにすぎない．単にとてつもなく早く物事を行なうことができるために，非常に泥臭く行なっていることにわれわれは気がつかないのである．(興味深いことに，脳のニューロン(神経単位)は基本操作を実行するのにおよそ数ミリ秒を要するので，脳がなぜこれほど賢いのかは謎である．コンピューターは掛け算をするときには脳よりはるかに勝るが，人を識別したり物を巧みに扱うという幼い子供にさえ簡単なことがコンピューターには困難なのである.)

さらに進んで，われわれは何が基本的な操作であるかをもっと明確にする必要がある．もっとも基本的な操作の1つは，事務員が読んだカードの情報をメモ帳に転送する作業である．

転送操作

「カードXを取り出す」＝カードX上にある情報をメモ帳に記入する

「カードYに書きもどす」＝メモ帳の情報をカードYに記入する

われわれが行なったのは，カードX上の情報をメモ帳に複写することを意味する「カードXを取り出す」という命令と，同様に「カードYに書きもどす」という命令を定義したことだけである．次に，事務員にカードXに書いてある担当地域が「カリフォルニア」であ

るかどうかを調べるように命令できるようにしたい．彼はこの作業を各カードについて行なわねばならない．そこで，彼がまずしなければならないのは，あるカードから次のカードの間で「カリフォルニア」を忘れずに覚えていることである．彼を助ける1つの方法は別のカードCに**カリフォルニア**と書いておくことである．そこで彼への命令は次のようになる

カードXを取り出す(保管場所からメモ帳へ)
カードCを取り出す(保管場所からメモ帳へ)
カードXに書かれている内容とカードCに書かれている内容を比較する

われわれは彼に，もし2つの内容が一致している場合にはこういうことを行ない，一致していない場合にはカードをもどして次のカードを取り出しなさいと教える．カリフォルニア・カードを取り出したりもどしたりしつづけるのはいささか非効率的に思われる．実際のところ，これを行なう必要はない．作業中はカリフォルニア・カードをしばらくメモ帳の上に置いておけばよい．これはより良い方法だが，すべては事務員のメモ帳のゆとりと彼がどれほど多くの情報量を保持する必要があるのかに依存する．もし，あまりゆとりがなければ，カードをもどしたり取り出したりを繰り返さねばならない．われわれはそのようなことを気にしなければならない！

事務員の複雑な作業をより簡単で基本的な作業に分けつづけることができる．たとえば彼に，保管場所から取ってきたカードの「担当地域」の部分をどのように見つけさせればよいのだろうか？　1つの方法はこの哀れな事務員に

0000　0000　0000　0000　0000　1111　0000　0000　0000　0000　…

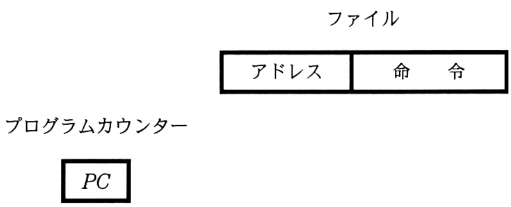
と書かれている別のカードを読ませることである．連続した数字のグループは，それぞれカードの特定の部分の情報に対応している．最初のゼロの組は，たとえばセールスマンの名前を表し，次の組は年齢，などとなっている．事務員は数字1の組に出合うまでこの数値リストを照合していき，見つけたらその隣の情報を読みとる．この例の場合は，1111はカリフォルニアを表している．この種の処理としてコンピューターでは，後で議論する「ビットごとのAND」操作が実際に使われている．少し回り道になったが，この事務員が特殊な技能を持っているのを当然と思う必要はないこと，そして彼にますます頭を使わずに仕事をさせられることを君たちに印象づけるための議論である．

1-2　命令セット

では，事務員のメモ帳を調べてみよう．われわれはまだ事務員にこれをどのように使うか教えていないので，ここで教えよう．彼が実行できる命令は2つのグループに分けられると仮定しよう．まず最初が，

たとえば加算や転送など，メモ帳にもともと備わっている核となる単純な処理の「命令セット」(一揃い)である．これらはハードウェアの中にある．われわれが問題を変えてもこれらは変わらない．これらは事務員の基本能力を反映しているといえる．次が，たとえばセールスマンの歩合の計算というような，ある作業に固有な命令の一揃いである．この一揃いに属する命令は，それぞれ核になる命令セットから前に議論した方法で構成され，その仕事をするために事務員に要求される能力の取り合わせを表している．

まず，われわれが事務員に行なわせねばならないのは，命令を正しい順序で実行させることである．われわれはメモ帳の1カ所を「プログラムカウンター(*PC*)」にすることでこれを行う．ここには番号が書いてあり，この番号は事務員が計算手順のどこにいるのかを示す．事務員にとって，この番号は命令の番地(アドレス)である．すなわち，ファイルシステムの中には特別な「命令ファイル」キャビネットがあり，プログラムカウンターに書かれている番号は，命令ファイルの中から彼が取ってこなければならないカードの番号である．このカードには彼が次に行なわねばならない命令が書かれている．そこで彼は命



令を受け取り，それをメモ帳の「命令レジスター」と呼ばれている領域に保存する．
この命令を実行する前に，彼はプログラムカウンターの内容に1を加えることによって次の命令のための準備をする．それから彼はレジスターの中の命令が彼に実行せよと告げるものをすべて実行する．一連の行為を次のように表すことができる*4．

PC に書かれているアドレスから命令を取ってくる
$PC \leftarrow (PC)+1$
命令を実行する

ここで括弧記号(～)は「～の内容」を意味する．これから何度も使うので覚えておくこと．第2行目はプログラムカウンター *PC* が新しい値 $(PC)+1$ を「得る」ことを示す凝った表記法である．このほかに算術演算などを行なうためのいくつかの一時的な記憶領域がメモ帳に必要である．この一時的な記憶領域はレジスターと呼ばれ，彼がファイルのところに行って他の数を探している間，何かを保存しておく場

*4 この「レジスター転送言語」の書き方は著者のものとは変えてある．われわれは標準的なプログラミング言語に使われているいわゆる「右から左へ」という書き方に従っている．[編者]

所である．単に2つの数を加える場合でさえ，第2番目の数を取ってくるまで最初の数を覚えておく必要がある！　すべての作業は順序どおりに実行しなければならないが，レジスターがこれを可能にする．普通，各レジスターには名前がある．これからの説明ではレジスターは4つあり，それぞれ A, B, X, C と呼ぶ．1ビット*5のデータのみを保存できる第4番目のレジスター C は特別で，これを「桁上げレジスター」と呼ぶ．レジスターの数はもっと多い場合もあるし少ない場合もある．一般的に言って，より多くのレジスターがあればそれだけプログラムを書くのが簡単になる．しかしわれわれの目的のためには4つで十分であろう．

このような次第なので，この事務員は何をいつ行なわねばならないかをどのようにして見つけるかを知っている．さて，彼のメモ帳に対する核になる命令の一揃いを調べてみよう．第1番目の種類の命令は1つのカードから別のカードへのデータの転送に関するものである．たとえば，メモ帳には記憶場所 M が書いてあるとする．われわれはレジスター A の内容を M に転送する命令が欲しい．

　　(A) を M に転送する　　すなわち　　$M \leftarrow (A)$

同様に，こんどは逆向きに，M の内容を A に書き込みたいと思うこともあるだろう．

　　(M) を A に転送する　　すなわち　　$A \leftarrow (M)$

ここで M は A のような一時的な記憶場所として設計されているとは限らない．レジスター B についても類似の命令が必要である．

　　(B) を M に転送する　　すなわち　　$M \leftarrow (B)$

　　(M) を B に転送する　　すなわち　　$B \leftarrow (M)$

レジスター X は少し違ったふうに使い，B から X への転送と X から B への転送を可能にする．

　　$X \leftarrow (B)$　　および　　$B \leftarrow (X)$

さらに，プログラムカウンター PC の中身を控えておいたり，操作したりできる必要がある．これは明らかに必要で，たとえば，もし事務員がある掛け算を実行するとしたら，彼がもどってきたときに次に何をするのかがわからねばならない．すなわち彼は PC の番号を思い出さねばならない．実際，われわれはレジスター X にその番号を保持する．したがって，われわれは次の転送命令をつけ加える．

　　$PC \leftarrow (X)$　　および　　$X \leftarrow (PC)$

次に，われわれには算術演算と論理演算が必要である．これらの中で最も基本的なものは「クリア」命令である．

　　A をクリア　　すなわち　　$A \leftarrow 0$

これは A に入っているものは何であっても，それを忘れ，消し去ることを意味している．次に加えるという演算が必要である．

　　B を A に加える　　すなわち　　$A \leftarrow (A)+(B)$

これはレジスター A が B の内容と A の以前の内容との和を受け入れ

＊5　2進数の1桁の数字(0または1)をビットという．[訳者]

ることを意味している．桁移動演算も必要である．これによって掛け算という核になる命令を導入することなく掛け算を実行することが可能になる．

　　A を左へ桁移動　　そして　　A を右へ桁移動

最初の演算は A のすべてのビットを単に1つだけ左に移動させる．もしこの移動によって左端のビットがオーバーフローすれば(あふれれば)，それを桁上げレジスター C に保存する．われわれは数を右に移動することもできる．私は今のところこの演算の使い道がわからないが，役立つことがあるかもしれない！

次の命令は論理命令である．これについては次の章で詳しく調べるが，ここでも完全を期すために言及したい．われわれの関心は AND, OR，そして XOR の3つである．これらは2つのディジタル「入力」x と y の関数である．もし**両方の**入力が1ならば AND は1になり，そうでなければ0になる．やがてわかるように，AND 演算は2進法の加算に現れ，したがって乗算に現れる．もし x と y を足し合わせる2つの数字として見れば，(x AND y)は両方の数字が1のときにだけ1になる桁上げビットである．レジスターによる表現では，x と y は(A)と(B)であり，AND はこれらに作用する．

　　AND：　$A \leftarrow (A) \wedge (B)$

ここで AND 演算子として論理記号∧を使った．一対の変数に AND のような演算子を作用した結果は「真理値表」に表せる(表1-1)．

表1-1　AND 演算子の真理値表

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

$X = A \wedge B$

他の2つの演算子も同じように表せる．OR もまた(A)と(B)に作用する．OR は，(A)と(B)の両方が0の場合以外は1になる．つまり，(x OR y)は x **または** y が1ならば1になる．XOR，すなわち「排他的 OR」は，(A)と(B)の両方が1のとき0になる以外は，OR に似ている．XOR は x と y の2進数の加算で，桁上げビットを無視した場合に対応している．1と1の2進法の加算は10になるが，この結果は桁上げビットを見ないことにすると0になる．OR と XOR に関連した論理記号を導入できる．

　　OR　　$A \leftarrow (A) \vee (B)$

　　XOR　　$A \leftarrow (A) \oplus (B)$

OR と XOR の作用も真理値表にまとめられる(表1-2)．

あと2つの便利な演算は，A の内容を1だけ増加させる(インクリメント)命令と1だけ減少させる(デクリメント)命令である．

　　A をインクリメント　　すなわち　　$A \leftarrow (A)+1$

　　A をデクリメント　　すなわち　　$A \leftarrow (A)-1$

表 1-2　OR 演算子と XOR 演算子の真理値表

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

$X = A \vee B$

OR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$X = A \oplus B$

XOR

明らかに，とても便利かどうかわからない命令を追加しつづけることができる．われわれはすでにいくつかの有用な計算をするために必要最小限以上の命令を持っている．しかし，われわれは可能な限り多くのことができるようにしたいので，他の命令も持ち込む．1つの有用な命令はデータ項目をレジスターに直接格納することを可能にする命令である．たとえば，**カリフォルニア**とカードに書いて，カードからメモ帳に写すよりも，**カリフォルニア**とメモ帳に直接に書き込めれば便利だろう．そこで，われわれは「直接転送」命令を導入する．

　　直接転送：　$B \leftarrow N$

ここで N は任意の定数である．

さらに追加すべき，きわめて重要な1種類の命令がある．それは分岐命令すなわちジャンプ命令である．「Z へジャンプせよ」は，事務員に対して(命令の入っている)場所 Z を見ろという命令である．普通はプログラムカウンターの値を1だけ増加させるところを，この場合はもっと大幅に増減させることを意味している．これによって事務員がプログラムのある部分から別の部分へ跳び移れるようになる．ジャンプには「無条件」および「条件付」の2種類がある．無条件ジャンプは上述した

　　Z にジャンプせよ　　すなわち　　$PC \leftarrow Z$

である．本当に新しい命令は条件付ジャンプ

　　もし $C=1$ ならば Z にジャンプせよ

である．この命令では，桁上げレジスター C が1の場合だけ，場所 Z へのジャンプが実行される．この条件付命令によって与えられる自由度はわれわれが関心を持つすべての機械の全体設計にきわめて重要である．

この他にも多くの種類のジャンプがある．プログラムの中のある決まった場所へのジャンプばかりでなく，あるステップ数だけ先の場所にジャンプできると便利なことがある．そこで，このステップ数をプログラムカウンターに加えるジャンプ命令を導入できる．

　　$(PC)+Z$ にジャンプせよ　　すなわち　　$PC \leftarrow (PC)+Z$

　　もし $C=1$ ならば $(PC)+Z$ にジャンプせよ

最後に，事務員にやめろという命令が必要である．

　　停止

これらの命令によって，したいことは何でもできるようになった．そこで以下に練習のためのいくつかの問題を示そう．その前に，われわれはどこにいて何をしようとしているのかをまとめておこう．これまでコンピューターの基本処理や手法を概観し，コンピューターの中には実際に何があるのかを示してきたつもりである(私は実際の構造は書かなかったが，設計のあらましは書いた)．簡単なコンピューターには少ししかレジスターがないが，複雑なコンピューターには多くのレジスターがある．しかし両者の設計概念は基本的に同じで，後者は少しスケールアップしただけである．

これまで考えた命令をどのように表すかを調べよう．われわれの場合に命令は，命令の対象アドレスと命令番号すなわち「演算コード」という2つの部分を含んでいる．

命令の対象 アドレス	命令 演算コード／番号

たとえば，命令の1つは「メモリーMの内容をレジスターAに入れる」ことであった．コンピューターは英語を話さないので，命令をコンピューターが理解できる形，言い換えれば2進数の列に記号化しなければならない．これが演算コードすなわち命令番号であり，その長さによってどのくらい多くの異なった命令が持てるかが明確に決まる．もし，演算コードが4桁の2進数であれば，$2^4=16$の異なる命令が持て，メモリーアドレスの内容をAに転送する命令はこのうちの1つである．命令のもう1つの部分は命令の対象アドレスであり，これはコンピューターがAに入れるべきものを見つけに行く場所，すなわちメモリーアドレスMがどこかを示す．「Aをクリアせよ」というようないくつかの命令はアドレスの指示を必要としない．

命令の演算コードがどのように表されるかとか，どのようにメモリーに並べられるのかといった詳しいことは命令を使うために必要でない．このようなことは一連の階層構造の最初で最も初歩的な段階である．われわれはこの段階を一貫して知らないままでいられるようにしたい．言い換えると，最初の人間が下層の詳細を考えて設計し，次の人からは下層段階の詳細について気にせずに，設計したものを使用できるようにしたい．

われわれが今まで完全に無視してきたことがある．これまで記述してきたわれわれの機械は数字を出し入れする手段がないので動作しないだろう．入力と出力を考えねばならない．これについての手っ取り早い方法の1つは，メモリーの特定の場所，たとえばアドレス17642

を入力に割り当て，このアドレスをキーボードに割り当てて機械の外部からそのアドレスの内容を変えられるようにすることである．同様に別の場所，たとえば17644を出力とし，そのアドレスをTVモニターまたは何か他の装置と連結させて，計算結果が外界に達するようにする．

さて，このような問題の理解を深めるには2つの方法がある．1つの方法は基本的なアイディアを覚えて家に帰り，どのような命令が必要かを考えてみて，必要な命令が1つ残らずあるのを確かめることである．命令の一揃いの大きさを便利になるよう適当に小さくしたり大きくしたりして問題を解いてみて，それぞれの場合の長所短所を理解するように努めなさい．これが私の性格に合った私のやり方である！　あることを解こうとすることによって理解する，言い換えれば，あることを考案することによって理解する．これが私の勉強法なのである．もちろん，100パーセント考案するのではない．しかし，考え方のヒントは見ても，詳しい解き方は覚えない．詳しい解き方は自分で考えるのである．

もう1つの方法は，これもまた価値のある方法であるが，他の誰かがやったことを注意深く読むことである．基礎知識を理解してしまった場合には，最初の方法が私にとって良い方法である．問題を解いて行き詰まった場合には，その問題の解き方が書いてある本を探して，ページをめくって「あ，これを忘れていた」と気づくと，本を閉じて，問題を解くことにもどる．その問題の解き方がわかった後で本に出ている解き方を読んで，自分の解き方の下手なことと，本の解き方がはるかに頭脳的で効率的なことを知る！　しかしこのようにして初めて彼らの考え方が頭脳的なことを理解でき，また問題を考える際の枠組を持つようになるのである．初めから他人の解き方を読み始めると退屈で面白くなく，全体像を把握できない．少なくとも私にとってはそうなのである！

この本では，君たちの練習のためにいくつかの問題を提示する．君たちは問題を解かないで先に進もうという気持ちになるかもしれない．もしその問題が難かしすぎるのなら，飛ばしても構わない．いくつかの問題はかなり難しい！　しかし，「この問題はおそらく誰かが前に解いているだろう．そのような問題を解くことに意味があるのだろうか？」と考えて飛ばすかもしれない．**もちろん**その問題はすでに解かれている！　しかしそれがどうした？　その問題を楽しむために解きなさい．そうすることが，いつか問題を解かねばならないときに，それを解くコツを覚える方法である．ひとつの例を出そう．級数

$$1+2+3+4+5+6+7+\cdots$$

の総和を，たとえば62まで，とりたいと仮定する．もちろん，どう

すればよいのかを君たちは知っている．しかし，子供が答を示されずにこの問題を出されたら…，どのように解くのかを考えるのは子供にとって**楽しい**経験である．この子供が大人になったとき，物事を発見することができるという自信を育てるし，この問題の解答がすでに発見されていても，そんなことは気にならないだろう．ある人が解けたこと，他の人も解けたこと，そして他の人たちが君を出し抜いたという事実を気にしてはならない．逆に何かを発見したことに楽しく興奮すべきである．この本の問題のほとんどは何回も解かれてきたものであり，多くの独創的な解が考案されてきた．しかし他人が解いた問題を解きつづけ，自信を得，楽しむために自分の解答を複雑にしていくと，やがて**誰も解いたことのない**問題を解いたことを発見する日がくるだろう！　これがコンピューター科学者になる道である．

私が経験した実例を紹介したい．いま，整数の総和の話をした．さて，何年も前に，私はこの問題の一般化に興味を持った．2 乗の和，3 乗の和，高次のべき乗の和を計算して，m 個の整数それぞれの n 乗の和までの公式を見つけようと努力した．私は，魅力のある多くの関係式を発見し，この問題を解いた．研究を終えたとき，私は各々の和に対する公式を見つけていた．この各 n に対する公式には数が 1 つずつ含まれていたが，私はこの数に対する公式を見つけることはできなかった．私はこれらの数を書き下ろしてみたが，これらの数を求めるための一般的な規則を発見することはできなかった．面白かったのはそれらの数が，$n=13$ になるまでは整数であったが，13 では整数でなかったことである(691 をわずかに越える値になった)．非常にショッキングで楽しかった！

後になって，私はこれらの数が 1746 年に発見されていたことを発見した．私は 1746 年の段階に到達したのだ！　それらの数は「ベルヌーイの数」と呼ばれていた．ベルヌーイの数に対する公式はとても複雑であり，普通の意味では未知だといえる．私は 1 つ前の数から次の数を求める「帰納関係」を得たが，任意の関係を見つけることはできなかった．このようにして私は，次に 1889 年に最初に発見された結果を発見し，次に 1921 年に発見された結果を発見し…，そして最後に私が発見した日と同じ日に発見された結果を発見した，というような経験をしてきた．この経験がとても楽しかったので，そのような経験をする人が他にも存在すべきだと考えている．こう考えて，君たちが楽しむために問題を出しているのである．（もちろん，一人一人は異なった風に楽しむのである．）君たちは過去の問題について多くの練習をしなければ何か新しいことを発見しそうもないが，それだけではなく，面白い関係や興味深い問題を調べることから多くの楽しみを得るべきである．また，他の人の研究を読めば，それを解くのがいかに難しかったか(またはそうでなかったか)，彼が何をしようとした

のか，何が問題だったのか，などがわかる．解答を読む前に問題を解こうといろいろ試してみた後の方が，理解するのがはるかに容易である．このような多くの理由で，私は君たちに問題を解くことを勧める．

問題 1-1　(a)　頭の悪いファイル事務員とカリフォルニアでの総売り上げを見つける問題にもどろう．君は，仕事を迅速に行なうために事務員を2人雇うことを経営者側に提言しますか？　もし提言するなら，2人をどのように使うか？　そして計算速度を2倍にスピードアップできるか？　事務員がどのようにして命令を受け取るのかを考えなさい．この解答を事務員が K 人の場合，あるいは $2K$ 人の場合に一般化できるか？

(b)　K 人の事務員によってスピードアップできるのはどのような種類の問題か？　明らかにスピードアップできないのはどのような種類の問題か？

(c)　最近のコンピューターのほとんどには中央処理装置が1つだけある．いわば，事務員が一人だけいる．その一人の事務員は一日中そこに座り，魔物のようにせっせと働き，カードを保管場所から死にものぐるいで出し入れする．機械の全体としての速度は事務員――すなわち中央処理装置――がこれらの操作をする速度によって決まる．どうすれば機械の性能を改善できるか調べてみよう．n が1024のような大きな数である場合，2つの n ビットの数を比較して，2つの数が同じかどうかを調べたい．1人のファイル事務員がこれを行なうのに最も簡単な方法は2つの数の各桁の数字を順々に比べることである．これに要する時間は明らかに桁数 n に比例する．さて，n 人，$2n$ 人，あるいは $3n$ 人のファイル事務員を雇ったと仮定しよう．何人の事務員を雇うかはわれわれしだいであるが，その人数は n の倍数でなければならない．ファイル事務員の数を増すと比較時間は $\log_2 n$ に比例するまで減らせることがわかっている．どうすればよいか分かるか？

(d)　もしこの比較問題ができたなら，もっと難しい問題に挑戦したいだろう．2つの n ビットの数を「$\log n$」時間で加える方法を考案してみなさい．この問題は桁上げを考慮しなければならないためもっと難しい！

問題 1-2　2番目の問題は事務員に掛け算をさせることに関する問題である(掛け算は基本命令の一揃いには含まれていない)．問題は2つある．まず，掛け算の実行のための適切な基本命令の一揃いを見つけなさい．掛け算をしたいときに毎回これらをコピーしなくても使えるように，これらの命令を機械のある場所，たとえば m から $m+k$ 番地に保存したと仮定しよう．事務員にこの命令の一揃いを使い，それからプログラムの正しい位置にもどるという命令をどのようにすれば与えられるかを示しなさい．

1-3 まとめ

これまでに，どのような機械の設計を理解していくのにも十分な話をした．しかし，特定の機械を詳しく調べるのではなく別のことをしよう．われわれは現在の位置から上，下または横に行くことができる．「上」とは多段階の抽象化を行なって機械の動作の詳細をユーザーに対してさらに隠すことを意味している．このような例はいくつか示した．たとえば基本命令から掛け算のような新しい演算を作ったのはその一例である．掛け算をしたいときはいつでもこの掛け算の「サブルーチン」を使えばよい．議論に値するもう1つの例は，メモリー内の位置ではなく代数変数を使えることである．X と Y の和をとりたいとし，その和を Z と呼ぶとする．

$$Z = X + Y$$

コンピューターは X と Y を知っており，メモリーの特定の場所に格納してある．われわれが最初に行なわねばならないことは Z の値を格納するためにメモリーのある場所を割り当て，それからこの場所が X と Y のメモリーセルの内容の和を保持するようにすることである．さてここまでくると Z についてすべてわかったので，Z を $Z+X$ のような別の式の中で使うことができる．変数について話す方がメモリーの場所について話すよりは明らかにはるかに簡単である．もちろん，こういう風にできるように準備するのは大変だが．これまでは数を転送するためにはその数がどこにあるかを正確に知らねばならなかった．いまでは，新しい数 Z を導入し，「私は数 Z が欲しい」，すなわち，「Z を入れる場所を見つけなさい，しかしそれがどこであるかを私には伝えてはいけない」とコンピューターに言うことができる．これが「上」に移動するという意味である．

1-2節で「A をクリア」などのような命令をまとめたときに，少し「上」に行ったことがあった．「A をクリア」のような略記はコンピューターのユーザーのために導入されていて，この略記を使って書かれているプログラムを機械が直接に理解することはできない．このような「アセンブラー言語」で書かれたプログラムはコンピューターが理解できる「機械語」に翻訳されねばならず，翻訳は「アセンブラー」と呼ばれるプログラムによって実行される．すぐ上の階層，つまり掛け算や変数などがあるような階層では，これらの「高水準」のプログラムをアセンブラー言語へ翻訳する別のプログラムが必要になる．このような翻訳プログラムは「コンパイラー」とか「インタプリター」と呼ばれている．この2つの違いは翻訳されるときにある．インタプリターは，順序に従って1行ずつ命令を機械語に通訳し，そのたびごとにプログラムを実行していく．コンパイラーはプログラム全体をまとめてアセンブラー言語か機械語に変え，そのあとまとめて実行する．ある場合には，コンパイラーは，プログラム全体を見通して必

要な諸操作を簡略化する巧妙な方法を見つけられるという利点を持っている．これは「コンパイラーの最適化」という重要な分野の核心であり，新しいタイプの「非ノイマン型」の並列コンピューターにとって重要性が増しつつある課題である．

新しいアルゴリズムを組み立て，いろいろな言語をプログラムし，プログラムやデータを含んだ「ファイル」を取り扱う能力を加えたりして上の階層に登りつづけることが明らかに可能である．現在ではほとんどの人が自分の機械をプログラムするのに高水準言語を使って高い階層で楽しく仕事ができる．機械語だけを使って仕事をすることがいかに退屈であったか想像してみてほしい！　そして現在でも，コンピューター設計者たちはそれをしているのである．

ここまでが「上」であった．下に行くときがきた．われわれの頭の悪いファイル事務員モデルと少数の命令リストより簡単なものがあるのだろうか？　われわれがこれまで考えなかったのはファイル事務員が何から**できている**かということである．もっと現実的に表現すると，これまで議論してきたさまざまな演算を実行するための電子回路をどのように組み立てるのかということである．これは次の章で取扱う問題である．その前に，「横へ」移動するとは何を意味しているのかについて述べよう．横へとはノイマン型アーキテクチャとはまったく異なるものを考えることを意味する．ノイマン型アーキテクチャは1つの中央処理装置(CPU)を持ち，すべての処理が「取り出し・実行」サイクルを通して入ったり出たりするアーキテクチャである．これとは異なる多くのもっと風変わりなコンピューターのアーキテクチャが現在実験中であり，いくつかは買うことができる機械として販売されようとしている．したがって「横へ」行くとは，詳しさについては同じレベルのままであるが，核心の構造が異なる機械によってどのように計算が実行されるのかを考察することを意味する．私はすでに，何人かのファイル事務員に協力して同じ問題を解かせるという問題でそのような「並列」計算機について君たちに考えてもらった．

第 2 章

コンピューターの機構

2-1 ゲートと組合せ論理

「下」への旅を，加算，転送，制御判断などいろいろの単純な演算を実行するのに必要なものを調べることから始めよう．これらの演算の**すべて**を実行するためには，ほんのわずかなものしか必要がないことがわかるだろう！　何が必要かという見当をつけるために「加える」という演算から始めよう．われわれの最初の重要な決断は，基数2で作業するようにすること，すなわち2進法を使うことである．2進法では数字は1と0だけである．われわれは1と0を「オン/オフ」の状態で電気的に表すので，2進法はコンピューターの枠組にやすやすとあてはまることがわかる．さて，2進数を表現するために，絵画的でとても簡単な技法を取り入れよう．2進数を1と0の列として書くかわりに，1つの数字に一区画を対応させることにして，製氷皿のトレイのような区切りのある細長いプラスチックとして見てみよう．もしある区画がからっぽならば，そこの数字は0であることを意味し，数字が1ならばそこに小石を1つ入れる．つぎにそのような2つの細長いプラスチックを取り上げ，これらが互いに加え合わされる数，すなわち「被加数」であるとする．答を入れるために，この2つの細長いプラスチックの下にもう1つ細長いプラスチックを置く（図2-1）．

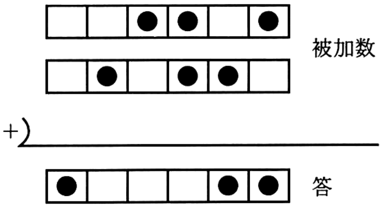


図 2-1　2 進数加算の絵画的表現

これによって抽象的な数学の問題が現実世界の「力学」の問題へと転化した．加算の実行に必要なのは小石の移動に対する一組の簡単な規則だけである．さて，遅くて取り扱いが面倒な小石の代わりに，他のもの，たとえば，1に対する高電圧と0に対する低電圧をとる導線を使うこともできる．基本的な問題は同じで，「小石あるいは電圧の組合せに対する規則は何か？」である．2進数の加算に対する基本的な規則は

$$0+0=0$$
$$0+1=1$$
$$1+0=1$$
$$1+1=0 \quad \text{および桁上げ}$$

である．そこで，とても頭の悪い人にどのように小石を動かすかを教える場面を想像してみよう．もし2つの小石がたてに並んでいたら，それらの下にある「和」の場所には小石を置かずに，「1つ左の区画に小石を1つ桁上げして置け」，などと教える．驚くべきことに，十分に詳しい規則を理解すれば，この頭の悪い人はどのような大きさの

2つの数でも加え合わせることができる！　もう少し詳しい規則の組があれば，彼は加算を卒業して掛け算にとりかかれる！　いつの日にか彼は，超幾何関数や君たちが使いたい方法を含むような非常に複雑なことさえできるようになる．小石を動かすことしかできない，見たところは完全に頭の悪い人に君たちが教える事柄は，彼が超幾何関数やその他の関数の数値を求めるのに十分である．もし彼が小石を十分に素早く移動させれば，彼は君たちより早く関数の値の計算を行なえることになる．この点で，彼が自分は君たちより頭が良いと思っても，それはもっともだ！

表 2-1　2進数加算の「真理値表」

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

もちろん，実際の機械は小石をもて遊ぶことによって計算するのではない(しかし昔のそろばんを忘れないように！)．機械は電気信号を手ぎわよく処理する．そこで，もしわれわれが考えている演算すべてをつくろうとすれば，電気回路についての考察を始めねばならない．製氷皿や石は捨てて，2つの2進数の数字 A と B を加える，現実的で，物理的な加算器を組み立てる問題を考えよう．この加算は和 S と桁上げ C という結果となる．これを表 2-1 のように表す．

加算器を2つの入力ライン A, B と2つの出力ライン S, C を持つブラックボックスとして表そう(図 2-2)*1.

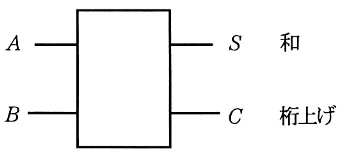


図 2-2　加算器のブラックボックス

このブラックボックスの実際の性質は，まもなく詳しく説明する．さしあたり，これが動作するものとしよう．(余談だが，2つの r ビットの数を加え合せるのにこのような加算器がいくつ必要だろうか？ $(2r-1)$個の単ビット加算器が必要なことを確かめなさい．この事実はまた，単純な装置から複雑な装置を系統的に組み立てる一般原理の別の例になっている.)

単ビット加算器というブラックボックスにもどろう．桁上げビットを見てみよう．これは A と B の両方が1の場合にのみ0でない．これはブール論理のいわゆる AND ゲートの働きと正確に一致する．このようなゲートは，2つの入力と1つの出力，そして出力が入力にどのように依存するかを示す「真理値表」を持つブラックボックスにほかならない．AND ゲートの真理値表と普通使われている絵記号を図 2-3 に示す．A AND B は簡単で，もし A が1で**かつ** B が1のときに，そしてこの場合にのみ，1になる．したがって，桁上げと「AND」は実際に同じになり，加算器の桁上げビットは導線 A と B を AND ゲートに入力すれば得られる．これまでこのゲートをブラックボックスとして記述してきたが，A, B，および C に対する値として実際の電気信号を使い，現実の材料を使って，どのようにすればゲートを作れるかはよくわかっている．そこでわれわれは実際の加算器を作る準備ができた．加算器の和のビット S はもう1つの論理ゲート「排他的論理和」，すなわち XOR ゲートによって与えられる*2.

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

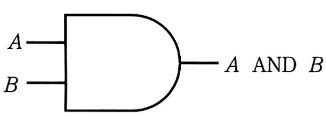


図 2-3　AND ゲート

*1　このブラックボックスは「半加算器」と呼ばれることがある．われわれはこの章で「全加算器」に出会うだろう．[RPF]

AND ゲートのように，XOR ゲートにも真理値表とかわいい記号がある(図 2-4)．*A* XOR *B* は，*A* と *B* の**両方ともではなく**，どちらかが 1 ならば 1 になる．XOR は似たタイプのゲートの OR ゲートとは異なっている．OR ゲートは図 2-5 に示した真理値表と記号を持っている．これらのゲートはすべて「スイッチング関数」の例である．これらは 2 進数の変数を入力としてとり，2 入力関数を計算する．シャノンは 1937 年に MIT での修士論文の中でブール代数の規則をスイッチングネットワークへ最初に適用した人物である．このようなスイッチング関数は「ゲート」(とてもぴったりした名前だ)と呼ばれる基本的な回路によって電気的に実現できる．導線に電気信号が現れるのが「1」(または「真」)で，現れないのが「0」(または「偽」)である．下の階層に降りていって，これらの基本的なゲートをもっと詳しく見てみよう．

いちばん簡単な演算は「恒等」すなわち「何もしない」演算である．これはある信号がブラックボックスに入ると同じ信号が出てくる 1 本の導線である．これは 1 つの導線を表しただけのものである(図 2-6)．実際のコンピューターでは，この素子は「遅延」素子と考えられている．第 7 章でわかるように，電流が導線を流れるとき実際には時間がかかり，機械を設計するときはこの有限の伝搬時間，すなわち遅延を考慮しなければならない．コンピューターでは，紙の上では何もしない素子でさえも，組み立てたときには何かができるのである！　この演算を飛ばして，2 番目に簡単な演算，すなわち入力信号を「否定する」ブラックボックスの考察に進もう．もし入力が 1 であれば出力は 0 であり，逆もまた成り立つ．これは理解しやすい真理値表(図 2-7)を持つ NOT 演算である．図式的には，NOT は頂点に丸印がついている遅延にすぎない．さて少し考えると，OR と AND には NOT を使った関係があることがわかる．真理値表を使うと，*A* OR *B* が NOT{(NOT *A*)AND(NOT *B*)} と同じであることが確かめられる．これは互いに等価な演算子の多くある例の 1 つである[*3]．もちろん，AND と NOT によって OR を表す必要はない．NOT と OR によって AND は表せる，などというだけのことである．論理ゲートを使って遊べる楽しいゲームの 1 つは，特定の目的に使うための論理ゲートの最良の組がどれで，この最良の組によって他の演算子をどのように表せるのかを見つけようとすることである．この問題を考えるときに自然に起こる疑問は，「**すべての可能な**論理関数を組み立てられるような論理ゲートの基本的な一組を揃えられるか，すなわち(起こりうるどんな入力状態に対しても 1 つの出力状態を割り当てることによって定義される)**どんな**ブラックボックスを考えても，その基本的な一揃いのゲートだけを使って**作れる**だろうか？」という疑問である．ここ

A	*B*	*A* XOR *B*
0	0	0
0	1	1
1	0	1
1	1	0

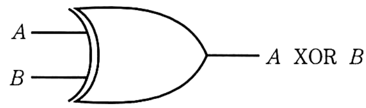


図 2-4　XOR ゲート

A	*B*	*A* OR *B*
0	0	0
0	1	1
1	0	1
1	1	1

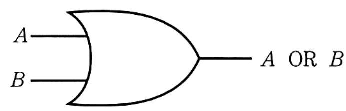


図 2-5　OR ゲート

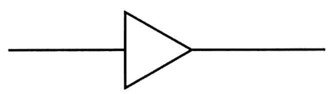


図 2-6　恒等

A	NOT *A*
0	1
1	0

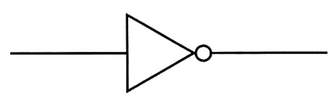


図 2-7　NOT ゲート

＊2　最近では XOR は EOR と記される．［訳者］
＊3　これらの関係は，ド・モルガンの定理として知られている一般的で由緒のある古い規則の特別な一例になっている．［編者］

ではこの一揃いの演算子の「完全性」の問題を詳しくは考えない．この証明はかなり手ごわいので，この講義のレベルを越えている．この章の2-4節での証明らしきものでがまんしよう．ここではAND, OR, およびNOTの組は完全な組**である**，すなわちこれらの演算子を使って，どのようなスイッチング関数でも必ず組み立てられる，と言えば十分である．この気のきいた問題を君たちがもっと考えたくなるように，それだけで完全な**1つの**演算子が存在すると書いておこう．

さて，技術者がいろいろなゲートを表すのに使っているかなり多くの記号がある．これらの記号は，対応する物理的な回路の間のつながりを図示するのに便利な道具である．たとえば，AND, ORおよびNOTの関係を図2-8のように図式で表現できる．導線に直接丸印を付けてNOTを表わすという一般的な書き方を採用したことに注意すること．三角形は描かない．

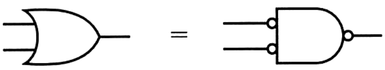


図2-8　AND, ORおよびNOTゲートの関係

これらの記号でしばらく遊んでみよう．XORはどのようにしたら作れるのだろうか？　XORは，$A=1$かつ$B=0$，または$A=0$かつ$B=1$のときにのみ1になる．このような新しいゲートを組み立てるための一般的な規則は，A AND B，A OR B，A AND(NOT B)などの真理値表を書き出し，望む結果が得られるように，あるゲートの出力を別のゲートの入力にする方法を見つけることである．たとえば，AとBをANDゲートに入れるとき，線BにNOTを付けておけば，$A=1$および$B=0$から1が得られる．同様に，$A=0$および$B=1$から1を得るために，今度は線AにNOTを付けたANDを使うという同じテクニックを使う．その上で，これらの2つのゲートの出力を3つ目のゲートのORに入力すると，最終的にXORが得られる(図2-9)．(われわれの描き方に注意すること．2つの交差した導線が電気的に接続されている場合には交点に黒丸をつけ，接続されていない場合には黒丸をつけない．)もちろん，このゲートの組み合せは，AとBの他の2つの入力の組み合せについても正しい出力を与えることを確かめねばならない．AとBの両方が0の場合には，両方のANDゲートは0になるのでORは0になり，AとBの両方が1の場合にも両方のANDゲートは0になるので最終結果は0になる．したがって，残りの2つの場合にもうまくいく．この回路が唯一の作り方ではないことに注意すること．XORスイッチを実現するもう1つの方法は図2-10のようなものである．

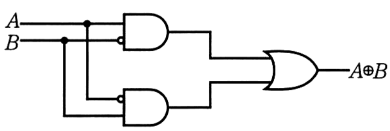


図2-9　ANDとORで表されたXOR

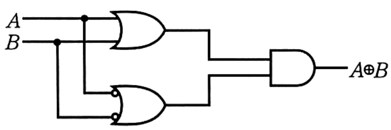


図2-10　もう一つのXOR

実際にはXOR回路をどの方法で作るべきなのだろうか？　それはハードウェア，半導体技術などの個別の状況の詳細に依存する．われわれは必要な素子の数が最小の方法はどれかというような問題にも興味がある．想像できるように，このような問題は興味深い設計の問題になるが，それについてはここでは深く立ち入らない．注意しなけれ

ばならないことは，AND, OR と NOT の入った大きな鞄があればどのようなスイッチでも作れるということだけである．たとえば，われわれはすでに単ビットの加算器を作るには，桁上げビットとして AND ゲートを，そして和のビットとして(AND と OR で作られる) XOR ゲートを使えばよいことを知っている．4 入力 A, B, C, D を持つ多入力 AND という別の例を調べよう．このゲートは 4 つの入力を持っているが出力は唯一つのままであり，2 入力の AND の場合を拡張して，4 つの入力のすべてが 1 のときにだけ信号を「発する」，すなわち 1 を出力するゲートであると定義する．この問題は記号で

$$A \wedge B \wedge C \wedge D$$

と表されることがある．ここで記号∧は(前に述べたように)命題論理における「AND」を意味する．もちろん，論理学者がこのような記号を書くときは，この演算を実行できる特定の回路を考えているわけではない．しかし，われわれは基本的なブラックボックスのゲートから作られた，そのような回路を設計できる．これは図 2-11 のような 3 つの AND ゲートの回路である．同様にして，どのような大きさの多入力 AND でも作ることができる．

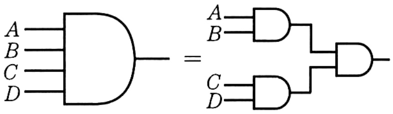


図 2-11　多入力 AND ゲート

さて，論理ゲートを組み立てるのに使う実際の電子部品を調べることによって，階層構造のほとんどどん底に到達するときがやって来た．われわれは実際にどん底まで降りる**つもり**だ．というのは，半導体の物理や機械の中の実際の電子の動きについて，このコースの第 7 章で議論するという意味である．さしあたり，エレクトロニクスをある程度知っていれば理解できるはずの，ゲートの組み立てについてのいくつかの手短な指針を与えよう．

すべてのゲート構造の中心は**トランジスター**である．トランジスターはすべての電子部品の中で最も重要であると言うことができ，工業の発展と成長において決定的な役割を果たした．トランジスターが含まれていない電子装置はほとんどなく，トランジスターがスイッチとして使われているコンピューターを理解するためには，トランジスターの基本的な性質の理解が不可欠である．NOT ゲートを組み立てるのにトランジスターがどのように使われるのかを見てみよう．図 2-12 の回路について考えよう．

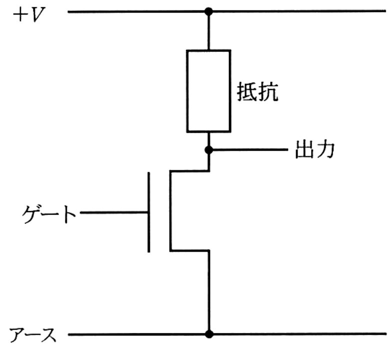


図 2-12　トランジスターの反転器，すなわち NOT ゲート

トランジスターは電極が 3 つある素子である．1 つの入力はゲート信号に，1 つは 0 ボルトにし，もう 1 つは抵抗を通って正電圧に接続されている．トランジスターの主要な性質は，ゲートが明らかに正電圧の場合には導通状態になるが，もしゲートが 0 または明らかに負電圧の場合には電流が流れないことである．次に，電圧をゲートに入力したときの出力電圧の振る舞いを調べよう．(規約として 1 と表す)正電圧を入力すれば，トランジスターの中を電流が流れ，出力電圧は 0

になる．これを2進数の0と表す．他方，ゲートが少しだけ負電圧，または0になったならば，電流は流れず，出力は$+V$と同じ，すなわち1である．こうして，出力は入力の反対になり，NOTゲートを手に入れたことになる*4．

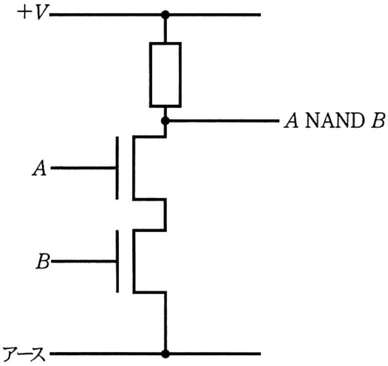


図2-13　トランジスターのNANDゲート

ANDゲートはどうなのだろうか？　トランジスターの性質によって，出発点としてNANDゲートを使うと便利であることがわかる．このゲートはMOSではANDゲートよりも作りやすいからである．NANDゲートを作ることができれば，ANDゲートはド・モルガンの規則の1つ，AND≡NOT{NAND}，を使うことによって得られる．そこで図2-13のような簡単な回路を考えよう．出力電圧が0になるためには，AとBの両方を通って電流が流れる必要があるが，これは明らかにAとBの両方が正電圧であるときにのみ実現される．したがって，この回路は確かに「NOT AND」すなわち「NAND」ゲートになる．ANDゲートを得るためには，図2-13のNANDゲートの出力を図2-12のNOTゲートに入力させればよい．この出力結果はANDゲートになる．

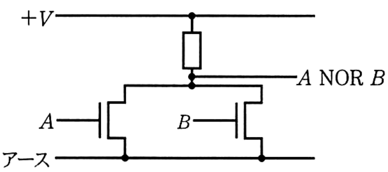


図2-14　トランジスターのNORゲート

ORゲートはどうだろうか？　すでにANDとNOTからORを作る方法を見てきたが，お望みなら上のようなトランジスター回路を組み合わせるというやり方で作ることができる．しかし，(概念的にも製造の観点からも)もっとやさしいORの組み立て法は，図2-14のような並列回路を考えると得られる．AまたはBが正電圧，または両方が正電圧ならば，電流が流れて出力はゼロになる．AとBの両方が0ならば，出力は$+V$，すなわち1になる．そこでこの場合にも，手に入れたいゲートの反対の機能を持つゲートの「NOT OR」すなわち「NOR」ゲートを手に入れたことになる．そこでこの出力をNOTに通すと，すべてがうまくいく．

これで君たちは基本的なゲートと同じ機能を持つ電気回路を作れることを確信したことと思う．次に複雑さの階層構造の1つ上のレベルにもどって，基本的な構成ブロックから組み立てられるもっと精巧ないくつかの装置を見てみよう．

2-2　2進数解読器

われわれが調べる最初の装置は「2進数解読器(デコーダー)」と呼ばれている．その動作は次のようである．装置に入る4つの導線A, B, C, Dがあるとしよう．これらの導線からは任意の入力が入る．われわれは導線の信号が特定の組み合せ，たとえば1011，であるかどうかを知りたい．すなわち解読器から1011が現れたことを報らせる信号を受け取りたい．これは解読器に入る4つのビットを調べ，それ

*4　技術的な余談として，われわれはこの回路をMOS(金属酸化物半導体)技術を使って組み立てると仮定している．この型のシリコン技術では抵抗を入れるのは難しく，実際にはこの抵抗を別の型のMOSトランジスターで代用する(第7章参照)．［RPF］

が 1011 であればわれわれに信号を送る超人を手に入れるようなものである．このためには AND ゲートを修正して使うのが簡単である(そして超人を雇うよりずっと安上がりである)．右の装置(図 2-15)では明らかに A, C, D が 1 で B が 0 のときにのみ 1 の出力が得られる．これは非常に特殊なタイプの解読器である．仮に，多くの起こりうる入力の組合せに対してそれぞれが自分自身に割当てられた特別な数を探す多数の超人がいるような，もっと一般的な解読器を手に入れたいとしよう．そのような解読器は個々の解読器を並列に接続することによって簡単に作ることができる．完全な解読器とは起こりうるすべての入力を解読するものである．このような解読器の動作を理解するために，3 入力 8 出力の 2 進数の解読器がどのように動作するのかを調べよう．ここでは，3 つの入力ビットとして導線 A, B, C が付けてあり，$2^3=8$ 種類の組合せの入力がある．したがって 8 つの出力の導線があって，どのような入力が解読器に入ったのかが一目でわかるように入力の組合せの各々を別の出力導線に割当てて，この 8 つの導線の 1 つだけに 1 を出すゲートを作りたい．この解読器は図 2-16 のように構成できる．ここで各水平線上にある 3 つの黒丸が，3 つの入力 AND ゲートを意味する，という描き方を導入した (図 2-11 のあたりの議論を参照)．各入力導線が A と NOT A の信号などに枝分かれしていることに注意すること．A が 1 ならば下の 4 つの導線だけに信号が送られ，A が 0 ならば上の 4 つの導線だけに信号が送られるように 3 入力 AND ゲートを配置した．同じように B と C (そして NOT B と NOT C)の導線上の黒丸は 8 つの出力のうちのどれに信号が送られるかを示す．各出力線の右側に対応する入力状態を記した．これで 3 入力 8 出力の 2 進数解読器を明示的に組み立てたことになる．

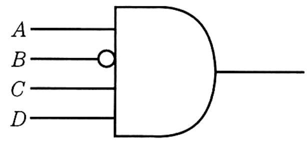


図 2-15　簡単な解読器

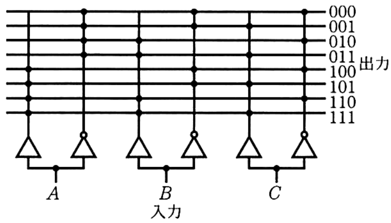
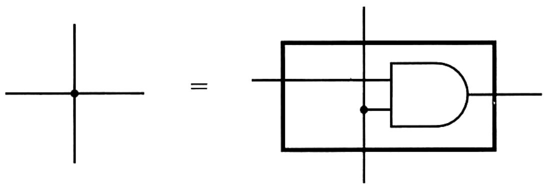
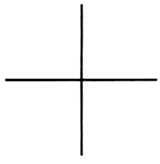


図 2-16　2 進数解読器

さて，図 2-16 に示した装置の解読器には，それが機械の設計者にとって必要不可欠な部品であることを示す重要な使い道がある．解読器のすべての水平な入力導線に左から 1 を入れることにしよう．交点の黒丸を 2 入力 AND として解釈し，



黒丸の付いていない交点では導線は接続されていないと解釈する．



左からの1の入力が最初の黒丸を通り過ぎるようにするために，交差する導線に $A=1$ または (NOT A)$=1$ のいずれかの正しい信号が現れていなければならない．B と C についても同様である．そこでこの装置は依然として2進数の解読器である．この点については何も変わっていない．しかし，われわれはまた，何か別のものを発明したのである．機能するコンピューターには絶対必要であることが，少し考えればすぐわかるようなものである．つまり，この装置は，入力導線のうち選ばれた1本に接続する多重スイッチとして使える．解読器のもともとの入力線，A, B, C はここでは，どの出力導線が信号(1か0)を出すのかを選択する「アドレス」線として使われる．これは「マルチプレクサー」と呼ばれるものにきわめて近いものである．マルチプレクサーとは複数の入力線からの信号を選択して一本の出力線に送る技術である．この例では，8入力ORゲートに8本の出力線をつなぐことによってこの装置を本物のマルチプレクサーにすることができる(図2-17)．このすばらしい複合装置は，3ビットのアドレス符号を使って，左の8本の入力線の中から送信すべき線を選択する．マルチプレクサーはコンピューターのメモリーへの読み書きをはじめとする多くの仕事に使われている．

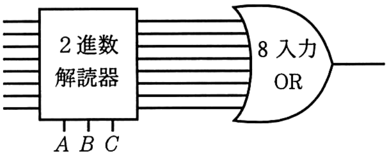


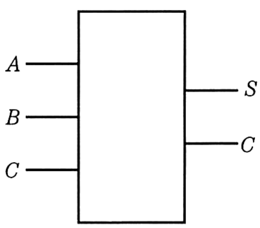
図2-17　マルチプレクサー

君たちが楽しむためのいくつかの問題を出そう．

問題2-1　8入力3出力の符号化器(エンコーダー)を設計しなさい．言い換えれば，前に考えた問題の逆の問題を解きなさい．すなわち8本の入力線があり，いつもそのうちの1本だけに信号が入っている場合に，どの入力線に信号が入っているかを3本の出力線に「符号化する」装置を設計しなさい．

問題2-2　AND, OR，およびNOTゲートを使って簡単な加算器を設計しなさい．

問題2-3　1ビットの全加算器（p.29を参照せよ[訳者]）を設計しなさい．



問題2-4　r 個の1ビットの全加算器を使って，r ビットの全加算器を作りなさい．どれくらいの数の1ビットの全加算器が必要か？

2-3　ゲートのつづき：可逆ゲート

ANDとNOTを組み合せた回路がすべてのスイッチング機能を実現するのに十分な構成材料になることは，証明は省いたが，前に話した．



われわれが気づかないうちに加えていた2つの別の素子がある．「分枝(FANOUT)」と「交換(EXCHANGE)」操作である(図2-18)．導線と1と0のパルスの場合，これらの「ゲート」の存在は誰の目にも明らかである．すなわち，分枝は1本の導線を2本以上の導線にただ分枝させ，交換は一対の接続の相手をただ交換するだけである．けれども，もし情報が小石によって運ばれるのならば，2つへの分枝は，1つの小石が2つになることを意味しており，これはとても特殊な操作である．同様に，もし情報が別の場所にある2つの箱に保存されていれば，交換も自明な操作である．われわれはこの2つの「自明な」操作が可逆性の議論で必要になるため，これらを含めることの論理的な必要性を強調しているのである．これ以外にわれわれが持っていると仮定するものは，0と1を無限に供給する供給源，すなわち導線を差し込めばそこから必要なだけの信号を取り出せる倉庫である．この倉庫には予期しない用途がある．たとえば，われわれは2-1節でANDとNOTのゲートの組み合せは1つのNANDゲートで置き換えられることを知った(図2-19)．このNANDゲートだけを持っていることはANDとNOTの両方を持つのと同じだということを理解するのは容易である．NOT演算をNAND演算から得るには，1の倉庫とNANDの1つの入力を接続しさえすればよい．このとき，他方の入力 A が何であっても，出力はNOT A になるからである．

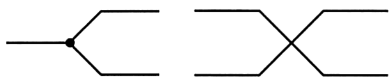


図2-18　分枝と交換

A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

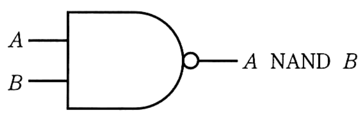
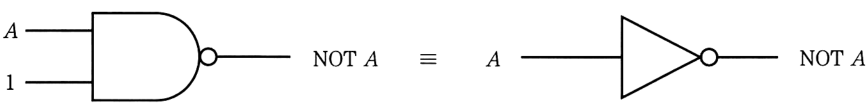


図2-19　NANDゲート



NOTとNANDがあるので，われわれは明らかにANDを作ることができ，したがってNANDはANDとNOTの組と等価であることが示された．

少し異なった問題を議論したい．これによって，いくつかのいくぶん風変わりな論理ゲートを調べられるようになる．ANDとNAND，そしてORとXORは**不可逆**演算である．不可逆演算とはゲートの出力信号から入力信号を再構成できず，情報は不可逆的に失われること

を意味する．もし4入力を持つANDゲートの出力が0ならば，15通りの入力の組のうちのどれか1つによる結果であるが，どの入力の結果であるかはわからない(もし出力が1ならば入力が何かは明確にわかる！)．**可逆**演算の概念を，入力を導き出すために十分な情報が出力に含まれている演算として導入しよう．この概念は後で計算の熱力学を学習するときに必要になる．これによって，計算の自由エネルギー(物理的効率といってもよい)についての計算ができるようになる．

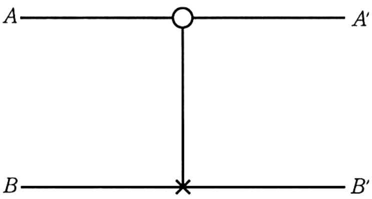


図 2-20　*CN* ゲート

表 2-2　*CN* ゲートの真理値表

A	B	A'	B'
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

可逆コンピューターの問題は，ベネットとフレトキンがそれぞれ独立に研究してきた．われわれが使う基本的な構成材料は，3つのゲート，NOT(N)，制御付NOT(CN)および二重制御付NOT(CCN)である．これらが何なのかを説明しよう．NOTは前と同じ単なるNOTで，1つの素子からなるものである．制御付NOTは2導線入力の素子であり，ANDやNANDゲートとは違い，2つの出力がある．これは次のように動作する．2本の導線があるが，そのうちの1本に円を書いて制御を表し，もう1本には×が書いてある(図2-20)．「×」印はNOT演算を表すが，このNOTは普通のNOTではなく，これは**○導線への入力信号によって制御される**．具体的には，○導線への入力信号が1の場合は，×導線への入力信号は反転される．○導線への入力信号が0の場合は，NOTゲートは動作せず，×導線の信号はそのまま通り抜ける．言い換えれば，○導線への入力信号は下の線のNOTゲートを活性化する．しかし，○導線の出力信号はいつも○導線の入力信号と同じであり，上の線は恒等演算になっている．このゲートの真理値表は簡単である（表2-2)．B'はXORゲートの入力AとBに対する出力，B'＝A XOR Bとして解釈できることに注意すること．

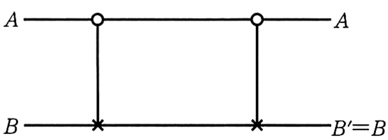


図 2-21　*CN* ゲートを使用した恒等

このCNゲートの最も重要な性質の1つは可逆性，すなわち，出てくるものから入ったものを推定できることである．このゲートの操作を逆にもどすには，このゲートの操作を繰り返せばよい（図2-21)．

CNゲートを使って分枝回路を組み立てられる．B＝0とすれば，B'＝AとA'＝Aが得られるからである．練習問題として，交換操作を行なわせるにはCNゲートをどのように接続すればよいかを示しなさい(ヒント：いくつかのCNゲートを使え)．

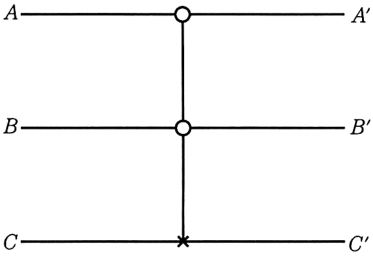
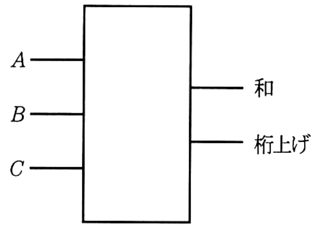


図 2-22　*CCN* ゲート

残念ながらNとCNゲートだけではすべての演算(操作)を行なうことができない．他のもの，たとえばCCN，すなわち二重制御付NOTゲート(図2-22)が必要である．このゲートには，2本の制御線AとBがあり，それぞれにはCNゲートの場合と同じ○印が付いていて，この線の信号はゲートを通過しても変化しない，つまりA'＝A，B'＝Bである．もう一本の線にはNOTが付いているが，これはA＝1かつB＝1の場合にのみ活性化され，C'＝NOT Cとなる．この

CCN ゲートは非常に強力であることに注意してほしい．A と B の両方を 1 にしている場合は，CCN ゲートはただの N，すなわち NOT になる．A だけを 1 にしている場合は，このゲートは B と C を入力として持つ単なる CN ゲートになる．したがって，CCN ゲートおよび 1 と 0 の信号源があれば，CN と N の両方のゲートを使わずにすむことになる．もっと良いことに，この CCN ゲートがあれば何でもできる！　われわれは CN ゲートに XOR ゲートの出力を出させるにはどうすればよいかをすでに知っている．AND ゲートは $C=0$ としておいて，入力として A と B をとることによって作れる．このとき，C' が A AND B の出力信号になるからである．というのは，C' が 1 になるのは明らかに NOT ゲートが活性化されて $C=0$ が反転されるときだけであるが，これは CCN ゲートの性質によって $A=B=1$ の場合だけだからである．したがって，明らかに，CCN ゲートだけですべてのゲートを作れるので，CCN ゲートはそれだけで演算子の完全な組である．

次にしなければならないのは，これらの可逆演算だけを使って何か便利なことができるのを示すことである．これは難しいことではない．それらを使って，演算子の完全な一揃いで実行できることは何でもできることをいま示したばかりである！　しかし，われわれが組み立てるものはすべて可逆的であってほしい．全加算器を作る問題を考えよう．



われわれは A, B，および C を加えて，和と桁上げを得る必要がある．さて一般に，和と桁上げから 3 つの入力信号を再構成することはできないので，現状のままではこの演算は可逆的ではない．われわれは可逆加算器が欲しいので，現在よりももっと多くの情報が出力側で必要である．ちょっと考えればわかるように，可逆ゲートは「入力する線の数」=「出力する線の数」という一般的な性質を持つ．これはすべての可能な入力が出力からそれぞれ「計算できる」ための唯一の方法である．そこでこの加算器から出ていく別の線が必要である．実際には，このゲートから出ていくあと 2 本の出力線と，入ってくるあと 1 本の余分な入力線が必要なことがわかる．この入力線を，たとえば 0 に設定する．N, CN および CCN(あるいは CN と CNN だけ)を使えば，AND, OR および XOR 演算子が得られるので，明らかにこれらを使って加算器を組み立てられる．このゲートを可逆にする巧妙な方法は，

和と桁上がりの他に2本の出力線を使い，それらが入力 A と B になるように配線することである．このゲートの配線を詳しく求めることは，取り組む価値のある練習問題である．

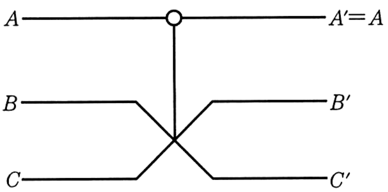


図 2-23　フレトキンゲート：制御付交換

フレトキンは，彼が考えるゲートの入力と出力にさらに制約を加えた．彼はゲートが可逆であることの他に，1と0の個数が変わらないことも要求した．この要求を課すことにはこれといった理由はないが，とにかく彼はそのように要求した．彼は制御付交換演算を実行するゲートを導入した(図 2-23)．彼の名誉のために，これをフレトキンゲートと呼ぼう．君たちはもう制御線という考えに慣れているはずである．制御線は他の入力に対する通常の操作を活性化するだけである．この場合には，操作は交換である．フレトキンのゲートは次のように動作する．$A=0$ の場合には，B と C は交換されず，$B'=B$ および $C'=C$ であるが，$A=1$ の場合には交換されて，$B'=C$，$C'=B$ になる．1と0の個数は保存されている．もっと難しい練習問題として，すべての論理演算を実行するために，*CCN* ゲートを使う代わりに，このフレトキンゲートが(おそらく驚くべきことに)使えることを示してほしい．

2-4　演算子の完全な一揃い

標準的な AND, NOT そして OR ゲート以外にもコンピューターにおける論理の問題があることを示すために，可逆ゲートという考えを紹介した．われわれは第5章で可逆ゲートの話にもどる．しばらくの間は可逆計算の話題から離れて，演算子の完全な一揃いの問題にもどりたい．さて，私はこれまで，演算子のいわゆる「完全な一揃い」があればすべての演算ができる，すなわち，どのような論理関数も構成できると述べてきた．完全な一揃いとして AND, NOT, FANOUT(分枝), EXCHANGE(交換)を取ろう．私が提示したい問題は，この一揃いが**完全だ**ということがどうすればわかるかということである．仮に n 本の入力導線の束があるとして，これらに $X_1, X_2, X_3, \cdots, X_n$ というラベルを付けよう．入力 $\{X\}$ の各パターンに対して，導線の組 $Y_1, Y_2, \cdots, Y_m$ へ特定の出力パターンがある．ここで m は n に必ずしも等しくはない．出力 Y_i は X_i の論理関数になっている．これを形式的に次のように書く．

$$Y_i = F_i(\{X\}), \qquad i = 1, \cdots, m \tag{2.1}$$

われわれが示したいのは，関数 F_i の**任意の**組について，入力に対してこの関数の働きをする回路がゲートの基本的な一揃いだけを使って組み立てられるということである．たとえば入力導線の和という例を調べてみよう．これを行なう方法は原理的には次のようであることがわかる．われわれの2進数解読器では n 本の入力導線と 2^n 本の出力導線があり，多くの AND ゲートを使って特定の出力導線が信号を出すように配線した．ここで，さらにこの出力が，他の出力導線の組に

特定の信号を出させるように配線したい．そうすると，出力導線の信号が特定の入力パターンの和の値に対応した2進数となるように配線することができる．

Xへの特定の入力の組として1本の導線を選び出し，この導線だけが「興奮」していて，残りのすべての導線は「冷静」な場合を考えよう．この導線が興奮しているとき出力信号の特定の組を生成したい．これは解読器の逆の問題である．われわれがいま必要としているものは**符号化器（エンコーダー）**である．君たちに示した問題の1つからわかったはずであるが，これは多くのORゲートから構成できる．われわれは問題を2つの部分に分けたことを君たちは知っている．以前に調べた第1の部分は，入力信号によって異なった導線が信号を出すようにするための配線法であった．これに対する解答はわれわれの解読器であった．われわれの符号化器は多くの入力導線を持っていなければならないが，同時には1つの入力導線にだけ信号が入る．われわれはこの信号を出した導線の番号を2進数で書き表せるようにしたい．3ビットの符号化器がORゲートから図2-24のように作れる．ここでORゲートに次のような描き方を使った．

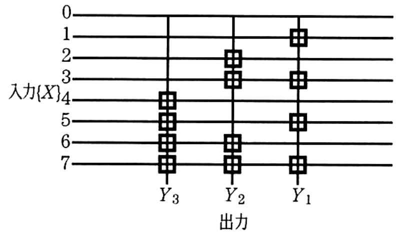
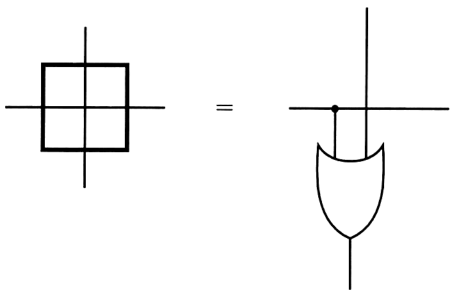


図2-24　3ビットの符号化器



このようにして，もし2^n本の導線に増えることを面倒がらなければ，われわれが望むすべての論理関数を構成できる．そのような装置には，一般に，AND平面とOR平面および2つの平面を接続する多数の導線がある（図2-25）．ここでANDゲートについては図2-16と同じ記法を使った．もし君たちが，いろいろな加算器を組み立てることを要求した問題2-2〜問題2-4のどれかを解くことに成功していたら，この原理のいくつかの簡単な例がうまくいくことがわかるだろう．

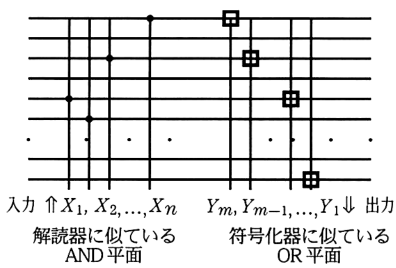


図2-25　一般的な論理関数の構成

この方法で構成できる論理関数のあるものはとても単純なので，ブール代数を使うと設計を単純化でき，より少ないゲートで実現できる．過去には人々は特定の論理関数に対する最も単純な，あるいは最も小さなシステムを見つけようと多くの努力を注いだものである．しかし，ここに記した方法はとても単純で一般的なので，設計には論理学の専門家を必要としない！　その上に，これはまたシリコンに容易に配置できる標準型の配置になっている．そこでこの型の設計は普通はプロ

グラム可能なロジックアレイ(PLA)に使われる．これらはコピーの必要が比較的少ない注文品のチップを製造するためによく使われる．顧客は望む機能を得るために，どのANDとどのORを接続するのかを指定しさえすればよい．大量生産のチップについては，配置をより効率的にすることは余分の努力を注ぐに値する．

2-5　フリップ-フロップとコンピューターのメモリー

さて，いままでとはちょっと違ったことを考えよう．これは便利なコンピューターの機能にとって中心的であるばかりでなく，調べるのも面白い．われわれは簡単な疑問から話を始める．「数字を記憶できるだろうか？」 つまり「これまで組合せてきたゲートや小さな部品からコンピューターのメモリーがつくれるだろうか？」 便利なメモリーとは格納しているものを変更できるもの，すなわち，メモリーの1つの場所の内容を消して書き換えられるものである．1つのビット(1つの1または0)だけを保持する，最も簡単なメモリーを調べてみよう．機能するメモリーを組み立てるときに，図2-26のようなブラックボックスをまず考えてみるのは合理的だろう．メモリーに何が入っているかを表現するために導線 C 上の信号を利用する．入力 A は制御線で，次のような性質を持つ．A が0である限り，つまりこのボックスに信号を何も入力しなければ，C は変わらない．しかし，A を1に変えると，C は0から1にあるいは1から0に変わる．この動作についてのある種の「真理値表」を書くことができる．この表から「次の C」は，A と現在の C のXORになることに気づく．そこで，頭を使ってこのブラックボックスを C からの**フィードバック**を持つXORゲートで置き換えれば，記憶装置になるかもしれないと思われる(図2-27)．これは作動するだろうか？ その答はすべてタイミング次第である！ われわれは抽象代数を解釈し，そして物理的な世界から装置に課される限界に注意しなければならない．A が0でかつ C が1だと仮定しよう．このときはすべてが安定している．ここまでは，うまくいく．次に入力 A を1に変えてみよう．何が起こるだろうか？ 定義によって，C は0に変わるが，これはわれわれが望んでいることである．しかしこの値はXORゲートにフィードバックされる．すると，$A=1$ なのでXORは1を出力する．そこで C は1にもどる．この C の値はXORにもどるが，$A=1$ なのでXORの出力は今度は $C=0$ になる．このようなことが繰り返されるので，このゲートはものすごく発振し，何の役にも立たない．

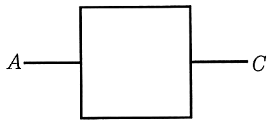


図2-26　メモリーのブラックボックス

表2-3　記憶装置の「真理値表」

A	現在の C	次の C
0	0	0
0	1	1
1	0	1
1	1	0

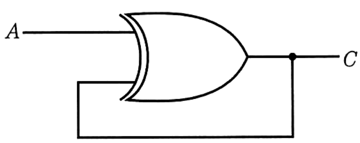


図2-27　もっともらしい(しかし作動しない)記憶装置

しかし少し考えれば，このゲートの演算のいろいろな段階に遅延を組み込むことによって，このゲートをある程度は救済できることがわかる．たとえば，XORが出力を出すのにある時間がかかるようにできる．しかし，発振を止めることはできない．短期間のメモリーを組み立てようとしても，電子部品の物理的状態の非持久性が余計な不安

定性を招いて，予期せぬ発振を生じ，このゲートを実用にはとても役立たないものにする．単に興味のためだけにだが「この回路を XOR ではなく OR で組み立てたらどうなるだろうか？」という問題を考えなさい．

明らかに，この装置の決定的にやっかいな特徴は**フィードバック**という要素である．これ無しで済ませることができないのだろうか？この答えはイエスであるが，それにはずいぶん費用がかかる．経済性とスペースの節約のために，コンピューターが同じ演算装置を使った繰り返し計算を行なえるようにしたい．たとえば，もしある計算の一部分を実行するためにある加算器を使ったなら，同じ加算器を使ってすぐ後の部分の計算もしたい．この計算には前の計算からの出力を使うかもしれない．われわれは，プログラムの新しい段階ごとにコンピューターの構造に多くの加算器を次々に積み重ねつづけたくない．しかしフィードバックを使わないと，そうするしかないだろう．そのためにわれわれはこの問題を解決したい．

われわれが欲しいのは，ある導線上の信号でリセットしようと決めるまで，0 または 1 の値を保持する回路である．この仕事をする回路は**フリップ-フロップ**と呼ばれ，図 2-28 のように図式的に描ける．フリップ-フロップには，「セット」(S)と「リセット」(R)という 2 本の入力導線，および Q と $\overline{Q}$ と呼ばれている 2 本の出力導線がある．後者のラベルには，一方がいつも他方の論理的な補数すなわち反転であるという事実が反映されている．これらは 0 の線および 1 の線と誤解させるように呼ばれることがある．誤解させるという理由は，他方が逆であるというだけで，それぞれが 0 と 1 のどちらの値も取ることができるからである．

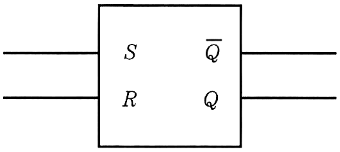


図 2-28　フリップ-フロップ

フリップ-フロップとして機能する回路を組み立てるために，たとえば，NOR ゲートを使うことができる(図 2-29)．この装置にはフィードバックが組み込まれていることに注意してほしい！　しかし，それにもかかわらずこのフリップ-フロップが，単純な XOR を使ったメモリーの場合とは違って発振しないように配線することができる．ただし，S と R が同時には決して 1 にならないように保証することが重要であり，これを保証するように機械の構造を整えることができる．この装置には $Q=1$（したがって，$\overline{Q}=0$）と $Q=0$（したがって，$\overline{Q}=1$）という 2 つの出力状態だけがあり，その両方とも安定である．この事実がメモリーにどのように役立つのだろうか？

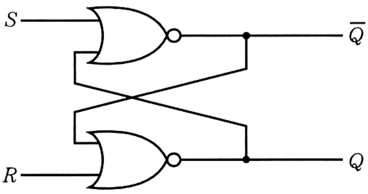


図 2-29　簡単なフリップ-フロップのゲート構造

この装置の動作はその真理値表(表 2-4)を調べることによってよくわかる．Q 線上の信号はフリップ-フロップの内容として解釈される．S と R が両方とも 0 の場合には Q の内容はいつでも同じ状態のまま

表 2-4　簡単なフリップ-フロップの真理値表

現在の Q	S	R	次の Q
0	0	0	0
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1
1	0	1	0

でいる．次にリセット線 R に信号が無い場合を考えよう．このとき，フリップ-フロップの内容 Q が最初に0の場合には，$S=1$ と設定すれば Q は1になり，他方 Q が最初に1の場合には S 線は Q に影響しないことがわかる．言い換えれば，S 線の信号はフリップ-フロップの内容 Q を1に設定するが，この後で S を操作しても Q は変化しない．すなわちフリップ-フロップ Q がすでに1の場合は，S を変えても Q の状態は1のままである．次にリセット線 R の影響を調べてみよう．フリップ-フロップ Q が0の場合には，$R=1$ と設定しても Q は0のままである．しかし，フリップ-フロップ Q が1の場合には，$R=1$ と設定すれば Q の値は0にリセットされる．したがって R 線はフリップ-フロップ Q の内容をクリアする．このフリップ-フロップを初めて勉強するときはかなりわかりにくいので，君たちがその動作を十分に理解するまでよく考えることを勧めたい．次にタイミングの問題を解決するためにこのフリップ-フロップがどのように使えるかを調べよう．

2-6　タイミングとシフトレジスター

さてわれわれは，フィードバックが組み込まれてはいるが，単純な構造の場合に起きるような発振に苦しむことのないフリップ-フロップを設計した．しかし，この装置には微妙で面白い問題がある．前節で指摘したように，いろいろな部品の間を信号が到達したり処理されたりするのには異なった時間がかかり，装置を組み立てるのに使っている部品の物理的な不安定さが，技術が理想的ならば生じない，ふらふらした変化をこれらの時間内にもたらすことがある．この事実は，信号が予定より遅れてゲートに到達して誤った仕事をすることを意味する！　われわれはこのような現象の影響がありうることを知っておかねばならない．たとえば，フリップ-フロップの両方の出力が同じになったら何が起こるだろうか？　われわれは理想化して，これらの信号が補数であると仮定したが，うまくいかないことがある！　このようなことが起こったら，セットやリセットといった仕事全部が，水泡に帰してしまうだろう．

これに対する方策はシステムに**クロック**(**時計**)を導入し，「イネーブル(enable)」信号を規則的な間隔でフリップ-フロップへ送ることである．そして，クロックの信号を受け取るまではフリップ-フロップが何もしないように工夫する．信号の間隔は，処理が実行される前にすべてが落ち着くのに十分なだけあいている．われわれは各入力導線に AND ゲートを置き，各ゲートにクロックの信号を送ることによってこの着想を実現する．この装置は**透過型ラッチ**と呼ばれることがある．その理由は，クロックが信号を出している間中ずっと，入力信号のどんな変化もこの装置を通って伝わるからである．

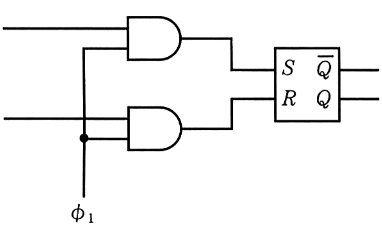


図 2-30　クロックを使った RS フリップ-フロップ

クロックからの信号ϕ_1をパルス信号列として表す(図 2-31). ANDゲートへの入力が何であっても，明らかにクロックからの信号ϕ_1が1のときにのみ入力信号はSとRへ伝わる．クロックのタイミングが適切で，かつ入力が不安定である可能性が無い状態になるまでゲートをオンにしないことを保証できれば，われわれはこの問題を片づけたことになる．しかしもちろん，われわれは別の問題を作ってしまった！　われわれは単にこの難題を後回しにしただけなのである．このゲートの出力が他のいくつかのゲートに入るので，伝搬時間などの同じ問題が再び起こるからである．すべてのゲートをクロックϕ_1に接続しても助けにならない．助けになるどころではない．システムのある部分は，別の部分の出力がちょうど変化しているときにオンに変わるかもしれない．まだ遅延があるのである．そこでこの困難を克服するために，遅延時間を計算し，すべてがうまくいくような高精度の機械を組み立てればよいと思うかもしれない．これは実現可能で，完成したシステムは速くて効率が良いだろう．しかし同時に非常に高価であり，設計するのが難しい．この問題を避けるための最善の方法は，もう1つのクロックϕ_2を導入し，このクロックがいいと言うまで，次に並んでいるゲートが前のゲートからの信号入力を受け入れないようにすることである．この取り決めは**マスター・スレーブ(主人と奴隷)型**フリップ-フロップと呼ばれる特殊な型のフリップ-フロップの基礎になっている（図 2-32). 2つのクロックの信号は相補的になっていなければならない．

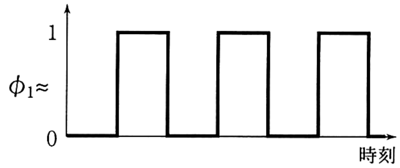


図 2-31　クロックのパルス

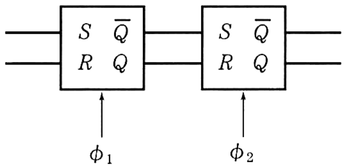
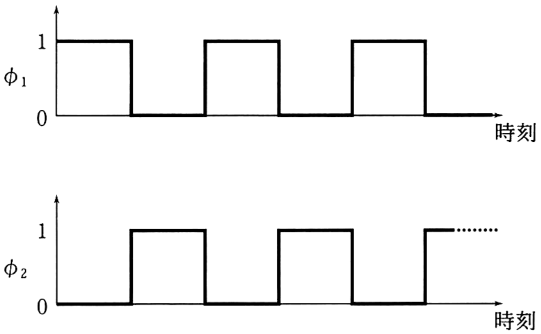


図 2-32　マスター・スレーブ(主人と奴隷)型フリップ-フロップ



相補性を保証する最も簡単な方法はNOT ϕ_1からϕ_2を得ることである．われわれはこのクロックパルスの長さに比べて論理演算を速くする必要があることにも気づく．これらの議論で次のような抽象化を使っていることを忘れてはならない．(1)信号の強さは1か0である(ほんとうは正しくない．信号の強さは時間とともにたえず変化している．決して厳密に1または0ではないが，近い値である)，そして(2)パルス間隔は一定で，一様である．われわれは**これ**が起こり，**あれが**起こるなどと言葉で言うが，これは理想化であり，望めばさらに多くのクロックからの信号を取り入れて理想に近づくことができる．

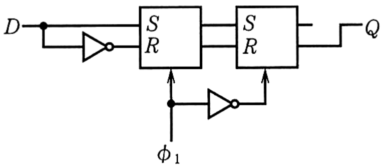


図 2-33　D 型フリップ-フロップ

いろいろなフリップ-フロップ装置を設計することが可能であり，それらがどのよう，にそしてなぜ機能するのかを学ぶことは取り組む価値のある練習問題である．そのような装置の1つは**D型**フリップ-フロップであり，図2-33に示す構造を持っている．なぜこの装置が「D型」フリップ-フロップという名前なのかは不明である．あるもっともらしい説明によれば「D」がこの装置の「遅延させる(delaying)」という性質に由来しているのだそうである．つまり出力は入力と基本的に同じであるが，クロックからのパルスの到着後にしかそうならない．

図2-34のようなD型フリップ-フロップの簡略表記法を導入しよう．

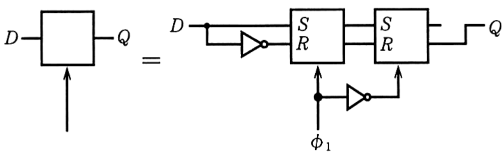


図 2-34　D 型フリップ-フロップの簡略表記法

フリップ-フロップから組み立てられる非常に役立つ装置は，これから考察する**シフト（桁送り）レジスター**である．この装置は，1つのビットだけでなく，任意の2進数，すなわち一連のビット列を格納できる装置である．この装置は直列に接続された多くのフリップ-フロップから構成され，われわれはこれに2進数を一度に1ビットずつ入力する．われわれは遅延が組み込まれている基本的な S-R を使う．シフトレジスターの基本構造は図2-35のようになっている．このレジスターの各単位は本質的には前に説明したD型フリップ-フロップのような安定な遅延装置である．並んでいるすべてのフリップ-フロップには同じクロック ϕ_1 が入っていることに注意すること．

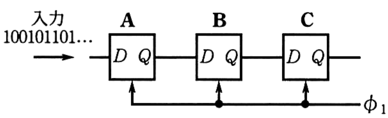


図 2-35　シフトレジスター

この装置がどのように機能するかを理解するのは難しくない．次のように仮定する必要はないのだが，簡単のために，すべてのフリップ-フロップが0にセットされていると仮定して始めよう．101という数をこの装置に入力してみよう．何が起こるだろうか？　まず最下位の桁の数字を入力するので，一番左側の A というラベルの付いている S-R に1を突っ込んで，クロックパルスが到着して信号を移動させるまで待つ．このクロックパルスの到着後に，A の出力は1になる．そこで次の桁の0を A に入力する．次のクロックパルスが到着するまでは何も起こらない．このパルスの到着後に，次の S-R である B は1を出力する(もともとの0はリセットされる)．しかし A の出力は，前方の入力を反映して0に変わる．その間に，この数の次の桁の1を A に入力する．ここでも，われわれは次のクロックパルスを待つ．このパルスの到着後に A の出力が1，B の出力が0，C の

出力が1になっていることがわかる——言い換えれば，左から右へ読むと，これはまさにわれわれがこの装置に入力した数になっている．もっと大きな2進数の列の場合に一般化するのは容易である(個々のフリップ-フロップは1ビットだけを保持でき，n 個のフリップ-フロップを含むレジスターは 2^n まで格納できる)．そこでこのようなレジスターは情報の断片の列を受け取り，それを1ビットずつ桁送りして並列の情報に変え，後の処理のために格納する．これ以上深入りすることは不必要だろう．このレジスターは明らかに数のメモリーや2進数の算術演算のための桁送り装置という用途があること，そしてその理由でこのレジスターは加算器や他の回路に組み込まれるということが読者にはわかるだろう．

第3章
計算の理論

これまでは，論理ゲートの構造によって課された計算の限界について議論してきた．この章では，「われわれが計算ができるものに，原理上限界があるだろうか？」というはるかに基礎的な問題について話をしよう．もし十分に大きなコンピューターを組み立てれば，われわれが望むものは何でも計算できると想像されるが，これは本当だろうか？　それともコンピューターがいかに立派に組み立てられていても，決して答えられないいくつかの問題があるのだろうか？

皮肉な事実であるが，この問題はコンピューターが作られる前から議論されていた！　ある意味では，コンピューター科学はコンピューターよりも前から存在していた．この話題は1930年代の論理学者や数学者にとって非常に大きな話題であった．当時，「何が**原理的に**計算できるのだろうか？」という問題について大騒ぎがあった．数学者たちはたとえばユークリッド幾何学のような公理と要素の数学的体系を組み立てて，これから何が演繹できるのかを調べるというゲームを楽しむ習慣があった．お決まりの仮定は，これらの数学的言語で書かれた言明は，どれでも成り立つか成り立たないかを原理的には証明できるというものであった．数学者たちはフェルマーの最終定理やゴールドバッハの予想のような，一見したところとても単純な言明を証明しようとして失敗をつづけていたが，遅かれ早かれ賢い男が現れて解答を発見するだろうと考えていた*1．しかし，やがてそのような命題が本質的に証明不可能なのではないかという疑問が生じた．この疑問は，論理学者のゲーデルが「ゲーデルの不完全性原理」で算術は不完全であるという驚くべき結果を証明したことから，深刻な問題になった．

3-1　有効な手順と計算可能性

何が証明できて何が証明できないのか，そしてどの数が計算できないのかを定義しようとする努力は，これから**有効な手順**と私が呼ぶ概

*1　フェルマーの最終定理の場合には，ある賢い男が解いた！　フェルマーの最終定理とは，方程式

$$x^n + y^n = z^n \qquad (n \text{ は整数}, \ n \geqq 3)$$

が整数の x, y および z について成立しないというもので，これまで数論の未解決の大問題の1つであった．この証明は不可能であると長い間信じられていたが(数学会はこれに懸賞金さえ出した！)，長い長い年月の研究(と1993年の虚報)の末に，ついに1994年に数学者のワイルとテイラーが成功した．［編者］

念へ導いた．有効な手順とは，各瞬間ごとに，ある目的を達成するために行なうべきことを示す規則の組，すなわちアルゴリズムである．例を使って，その意味をおおざっぱに説明しよう．君たちが数 x の指数関数 e^x を計算したいと思ったとしよう．これを実行するとても直接的な方法がある．テーラー級数

$$e^x = 1+x+(x^2/2!)+(x^3/3!)+\cdots \tag{3.1}$$

の使用である．x に値を入れて各項を合計すると，e^x が得られる．総和に含める項数を増やすにつれて，e^x に対する値は真の値に近づく．そこで，君たちが決めた課題が e^x をある精度まで計算することであれば，私は君たちにどのようにすればよいか教えられる．この方法は遅くて手間がかかるし，もっと効率的な技法があるのかもしれないが，私は気にしない．この方法はうまくいくからだ．これが有効な手順と呼んでいるものの一例である．

数学における有効な手順のもう１つの例は微分である．変数 x のどのような関数を選んでも，君たちが微分計算の基本規則を習得していれば微分できる．少しごちゃごちゃするかもしれないが，試行錯誤なしで素直に計算できる．これは逆演算の積分とは対照的である．だれでも知っているように，積分には技巧が必要である．与えられた被積分関数を積分する前に「変数変換をすべきか？　ある関数の導関数をその関数自身で割った形になっているか？　部分積分をするのが良いのだろうか？」などといろいろ推測しなければならない．この意味では正解へ直結する道をだれも知らない．われわれは積分について有効な手順を持っていないというのが公正な表現である．しかし，これはそのような手順が存在しないと言っているのではない．この分野における過去20年間での最も興味のある発見の１つは，そのような手順が**存在する**ということである！　具体的に言うと，正弦関数，指数関数，誤差関数などのあらかじめ定義されている初等関数で表されるすべての積分は，有効な手順によって計算できる．この事実は機械が積分できることを意味している．このことについて，われわれはリッシュさんに感謝しなければならない(The Problem of Integration in Finite Terms, *Trans. A.M.S.* 139(1969) pp.167-189).

数学には有効な手順がない別の例がある．たとえば，一般の代数式の因数分解である．4次までの式には有効な手順があるが，5次以上にはない．もし有効な手順があれば，すべての生徒がいくら払っても惜しくないと考えるであろう科目の興味深い例は幾何学である．幾何学の証明は，積分のように，すべての人に科学というよりむしろ技巧という印象を与え，かなりの発明の才が要求される．ところが，幾何学にも，積分のように，有効な手順が存在するのは皮肉である！　この有効な手順は直交座標の解析幾何学である．点の座標(x, y)を導入すると，ピタゴラスの定理や他のいろいろな公式を使って，すべての

距離や角度を決定できる．解析幾何学は，ユークリッド幾何学を代数学の一分野に還元する．それは有効な手順が存在するレベルである．

私は問題を有効な手順へ変換することと，その問題をコンピューターが扱うことができる形へ変換することとは，ほとんど同等であることを指摘した．そしてこれが，この話題が最近非常に注目を集めている理由(そしてたとえば，積分における有効な手順という考え方が最近になって初めて話題になり解かれた理由)の 1 つである．ところで，数学者が最初にこれらの問題を話題にしたとき，彼らの関心は計算の実際的な限界よりももっと一般的なものであった．すなわち，彼らは何が原理的に証明できるのかに関心があった．この問題はいろいろな研究法を生んだ．イギリスの数学者チューリングは「計算可能性」という概念をある型の計算機が計算を実行する能力と同等であるとみなした．チャーチは論理と命題の体系を定義し，これを有効な計算可能性と呼んだ．クリーンはいわゆる「一般帰納的命題」を定義し，これらを使って研究した．ポストはまた別の方法を使った(この章の最後の問題を参照のこと)．この問題を調べるのに別のいくつかの方法がまだあった．これらの研究者は皆，数学的言語といえるものを使って研究を始め，この言語の範囲内で「有効計算可能性」という概念を定義しようと試みた．ありがたいことに，これらの方法は見かけは異なっているが等価であることが示されている．したがって，われわれはこれらの方法のうちの 1 つを調べるだけでよい．われわれは最もよく知られているチューリングの方法を選ぶことにする．

チューリングのアイディアは，1 組の規則に従わねばならない数学者のような機械を作ることであった．彼の考えは次のようである．この数学者は正方形に分割された細長い紙を持っていて，その各々の正方形に一度に 1 カ所ずつ読み書きができる．彼がある正方形を見ると，彼の見たものが彼を，次の正方形に書くものを決定づけるある精神状態にさせる．そこで，この男の脳には多くの異なった互いに関係のある状態があり，紙片を見ると変化すると想像してほしい．こういう風に考えていき，若干の抽象化を行なって，チューリングはチューリング機械と呼ばれるまことに驚くべきある種の機械を提案した．この機械は，楽しみが目的でなければ誰もこれを組み立てることに時間を浪費しようとはしないほど，恐ろしく非効率で遅いが，それを我慢すれば，素晴らしいことができることがわかる．

さて，チューリングはすべての様式のチューリング機械を発明したが，最終的に最良のチューリング機械であるいわゆる万能チューリング機械(UTM)を発見した．UTM は，あらゆる特定目的のチューリング機械ができるすべてのことを実行できる．さらにチューリングは，**有効な手順で実行できるすべてのことは彼の万能機械によって実行で**

きるし，この逆もまた成り立つと主張した．この主張の対偶から，もしUTMが問題を解けなければ，この問題に対する有効な手順がないことになる．この主張は単なる推測にすぎないが，UTMと有効な手順に関するこの主張は広く支持され，多くの理論的な支持を受けた．だれもまだ理論上，UTMをしのぐ機械を設計していない．私は後でUTMの設計図を実際に示す．その前に，UTMのもっと単純な兄弟である有限状態機械を詳しく調べよう．

3-2　有限状態機械

典型的なチューリング機械は，テープと機械本体の2つの部分からできている．テープは必要とあればいくらでも長くできるものでなければならない．そして機械本体はテープの上を移動し，その内容を操作する．テープを非常に賢い機械の重要でない付属物であると考えるのは間違いである．テープがないと，この機械は実に無能である(頭の中で複雑な積分を解こうとしてほしい)．われわれは**有限状態機械**とよばれているテープの無いチューリング機械を調べることによって，チューリング機械とそれが実行できることについての検討を始めよう．

われわれは主としてチューリング機械の構成部分としての有限状態機械(FSM)に関心があるが，FSM自体にもいくらかの関心がある．FSMはどのような種類の問題を解けて，どのような種類の問題を解けないのだろうか？　FSMには答えられないがチューリング機械は答えられるいくつかの問題があることがわかる．なぜそうなのかという理由は当然，われわれの関心を引く．われわれは，この機械全体を，中の機械的な機能は隠されていて外からは見えない，ブラックボックスだとする．われわれは機械内部の詳細に関心がない．ただその動作にのみ関心がある．君たちが関連する概念に慣れるように，有限状態機械の例を示そう(図3-1)．基本的な考えは次のようである．この機械は運転開始時に**内部状態**Qにある．この状態は，たとえば単にある数をメモリーに保持しているというようなことである．それから機械は**入力**，すなわち**刺激**Sを受け取る．君たちはこの機械が(有限の)テープから1ビットの情報を読み取るかまたはこの情報をある別の方法で入力すると想像してもよい．この機械は入力に反応して**別の状態**Q'**に変化**し，入力Sに対する**応答**Rを吐き出す．機械の変化後の状態と機械の応答は，初期状態と入力の両方によって決まる．それから，この機械は別の入力を読み込み，再び状態を変え，再び応答を出すという一連の過程を繰り返す．

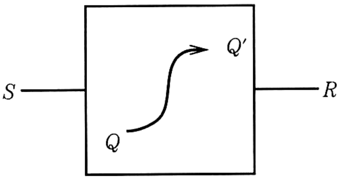


図3-1　一般的な有限状態機械

実際の機械を扱うために，離散的な時間変数を導入するが，この時間変数はすべての動作のペースを設定する．ある時刻tに，状態$Q(t)$の機械が記号$S(t)$を受け取る．この状況に対する応答が1パルス後の時刻$(t+1)$に生じるように機械を調整しよう．FSMを記述す

るために表記上2つの関数 F と G を導入し，

$$
\begin{aligned}
R[t+1] &= F[S(t), Q(t)] \\
Q[t+1] &= G[S(t), Q(t)]
\end{aligned}
\tag{3.2}
$$

と書こう．われわれはFMSの動作を適切な図法で描くことができる．機械がとることのできる状態の集合を $\{Q_j\}$ としよう．この機械が刺激 S を受けて，ある状態 Q_j からある状態 Q_k へ遷移し，その結果として応答 R を出すことを図3-2のように表す．この図法は多重刺激，多重応答そして多重状態変化の可能性があるときに真価を発揮する．たとえば，図3-3に示すようなシステムがあるとしよう．このFMSは次のように動作する．このFMSが，状態 Q_1 にあって刺激 S_1 を受ける場合は R_1 を吐き出して状態 Q_2 になる．しかし，これが刺激 S_2 を受ける場合には，R_2 を吐き出して状態 Q_3 に変わる．刺激 S_3 を受ける場合には，状態 Q_4 に移り R_3 を出力する．いったん状態 Q_2 になり，刺激 S_1 を受ける場合には，状態 Q_1 にもどり，R_2 の応答をするが，他方これが刺激 S_2 を受ける場合にはその状態のまま R_1 を吐き出す．読者はこの機械が状態 Q_3 と Q_4 にある場合には何が起こるかがわかり，もっと複雑な例を自分自身で構成できるだろう．

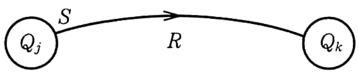


図3-2　状態遷移の図示

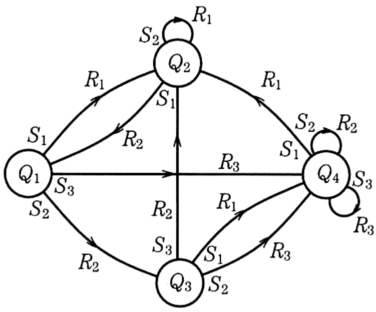


図3-3　複雑な有限状態機械

この例の1つの特徴は，この機械が3つの異なる刺激に応答できることである．これからさきでは，われわれの目的に合うように，可能な刺激を2進数の1と0の2つだけに限定する．この制限はFSMの実行内容には実際には影響せず，実行速度だけに影響を与える．1と0の数字列を与えることによる多重刺激も許される．これは明らかに2進数形式でなければならないが，任意の数を与えることと等価である．この種類の単純化は，処理速度に関心を払わないFSMやチューリング機械の研究ではよく行なわれる．

さて，遅延機械という，かなりつまらないものではあるが，実際に何かを実行するFSMの具体例を示そう．この機械に刺激を入力すると，ちょっと間をおいて同じ刺激で応答する．これが遅延機械の動作のすべてである．図3-4に遅延機械の「状態図」を示す．これより簡単な機械を作ることは難しい！　この機械には2つだけの内部状態があり，遅延機械として動作する．なぜなら，パルス化した時間を使い，時刻 t の刺激に対する機械の応答が時刻 $t+1$ に出てくることを要求しているからである．もしわれわれがこの機械に入力したものをすべて吐き出すように命令すれば，1単位時間の遅延がある．この遅延時間を長くすることは可能だが，もっと複雑な機械が必要になる．練習問題として，2単位時間の過去までを記憶する遅延FSM，すなわち時刻 t に入力した刺激が時刻 $t+2$ に出力される遅延機械を設計しなさい．(ついでに言えば，このような機械は，1単位時間だけの記憶を持つと考えることもできる．これに気づけば，時刻 $t+1$ での**状態**が時刻 t の入力を知らせていることになる．FSMの応答よりもその

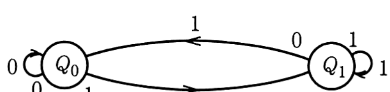


図3-4　遅延機械

状態を調べるほうが便利なことがよくある.)

FSMの働きを記述するもう1つの方法は前に記した関数FとGを表にすることである. このような表からFSMの動作を理解することは状態図から理解するよりも難しく, 非常に複雑な機械の場合には絶望的になるが, われわれは完全を期すために表3-1に示す.

表 3-1 一般的なFSMの状態表

G	Q_0	Q_1
S_0	Q_0	Q_1
S_1	Q_1	Q_0

F	Q_0	Q_1
S_0	R_0	R_1
S_1	R_0	R_1

さて, 状態図を使ってできることは驚くほどである. そこでその価値が評価できるように, 状態図の解読に習熟することは価値のあることである. 遅延機械より少し難しい例をもう少し示そう. まず最初は,「トグル」あるいは「パリティ」と呼ばれる機械である. この機械に0と1の数字列を入力すると, この機械は, 受け取った1の個数が偶数か奇数か, すなわちこの数字列のパリティを絶えず正しく示しつづけている. 図3-5から, 最後の数字を入力すると1単位時間後に, FSMは応答してパリティを知らせることがわかる. もし応答が1ならば, パリティは奇である, すなわち, 君は1を奇数個入力した. 応答0は君が1を偶数個入力したことを知らせている. 応答を見る代わりに, 機械の状態からもパリティが読み取れることに注意すること. そのために, 図の中の2つの可能な状態に私は「奇」と「偶」という記号をつけた.

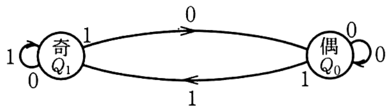


図 3-5 パリティ機械

いくつかの簡単な問題を出そう.

問題 3-1 1と0の数字列, すなわち2進数をある機械に入力すると仮定しよう. この数字列の数字の2つずつの和を計算する機械, すなわち, 入力数字を一度に2つとり, 2つを加え合わせ, その結果を2段階で出力する機械を設計しなさい. 2つの入力数字が00ならば, 00を吐き出し, 10または01ならば出力は01になる(1+0=0+1 !). 11ならば出力は10になる. 10進数では1+1=2だが, 2進数では10だからである. ヒントを与えよう. この機械には4つの状態が必要である.

問題 3-2 もう1つの質問は, 別な遅延機械の設計である. 今度は2つの入力刺激を記憶し, それを送り返す遅延機械である. この装置には, 4つの可能な入力00, 01, 10および11に対応する4つの状態が必要なことがわかるだろう.

問題 3-3 最後に, もし君たちがとりわけ元気があると感じているならば, 2入力の2進数加算器を設計しなさい. 私は完全な動作を要求している! 2つの2進数を, **最下位の桁を最初にして**, 同時に一度に1ビットずつ入力すると, このFSMはある遅延の後に和を出力するような加算器である. 私はこの加算器が和(S)だけを知らせてくれることに関心があり, 桁上げ(C)には関心がない. この機械の望ま

しい動作を図式的に描くと，次のようになる．

時間　⟶

入力　101011

　　　011010

出力＝和　110100　(1を次の欄に桁上げ)

3-3　有限状態機械の限界

もし君たちが加算器の設計に成功したなら，任意の大きさの2つの数を加え合わせられる単純な機械という少し驚くべきものを創造したのである．この機械は遅くて非効率だが，仕事はする．これはFSMではありふれたことである．しかし，このような機械の限界，特に，これらの機械が実行できない多くの仕事の存在を認識しておくことが重要である．実行できないことが何であるかを調べてみるのは興味のあることである．たとえば，任意の2つの2進数を掛け合わせるFSMを組み立てられないことがわかる．その理由はもっと簡単な次の例を調べた後で明らかになる．**括弧検査器**を組み立ててみよう．括弧検査器とはどのようなものなのだろうか？　次のような括弧の流れ

(((()) ((()()()((()) ()()(())))())

がある場合，括弧検査器の仕事はこのような表現が「つり合っている」，すなわち表現全体を通して括弧が矛盾なく開閉しているかどうかを確かめることである．これは左と右の括弧の数をただ数え上げて，両者が等しいことを確かめるのと同じではない！　左と右の括弧は正しい順につり合っていなければならない．これは，算術や代数学で，ほかの演算の中に入れ子の演算がある場合に共通の問題である．ところで，上の例は正しくない．次の例，

()(()())((()()(()())))))

は正しい．君たちは2つの場合の正誤を確かめておくこと．

一見，括弧検査器の組み立ては簡単なように思われる．簡単ではあるが，FSMでは検査を実行できない．検査の進め方の一例は次のようである．括弧の列の左端から出発し，閉じ括弧に到達するまで開き括弧の数を数える．閉じ括弧を左隣の開き括弧と「相殺」し，右に1つ移動する．閉じ括弧に出会えば，別の開き括弧でこれを相殺する．開き括弧に出会えば，相殺していない開き括弧の数に1を足して次の括弧に移動する．これは非常に簡単な方法であり，括弧の列がOKかどうかを知らせる．もし右端まで進んだあとで括弧が残っていれば，この括弧列には矛盾がある．それではなぜFSMはこのように簡単なことができないのだろうか？

その答は，われわれの望んでいる括弧検査器が**任意の**括弧列に対応できるものでなければならないという事である．つまり，一般にいくらでも多くの「開き」括弧を含んでいる任意の長さの列という意味で

ある．さて，この機械の本質的な特徴の 1 つが，処理の各段階で，何個の開き括弧が閉じ括弧と相殺されずに残っているかをずっと追いつづけなければならないということを思い出してほしい．しかもこれを実行するために，FSM の用語では，開き括弧の個数ごとに別の状態が必要である．ここに問題がある．**任意の**括弧列ということは任意の個数の状態，すなわち究極的には，無限の状態を持つ機械を要求するからである．したがって，どのような**有限**状態機械も任意の括弧列すべての検査はできない．括弧列検査ができるのは，おわかりのように，チューリング機械である．その理由は，チューリング機械は本質的に無限のメモリー容量を持つ FSM だからである．

私があら捜しをしていると思う人たちに対して，私は原理的な事柄を議論しているのだということを繰り返し述べておくことが重要である．実際的な観点からは，明らかに，遭遇しそうな任意の括弧列の整合性を検査する有限状態機械を組み立てることができるからである．機械の状態の数を決めれば，その機械が受け付けられる長さの括弧列だけしかこの機械に入力しないように保証できる．もし 32 ビットの 2 進数 32 個によって各状態を分類すれば，2^{1000} 個以上の状態を列挙でき，したがって，長さが 2^{1000} 個の括弧列が扱える．これはわれわれが実際に遭遇しそうな括弧列よりもはるかに長い．比較のために，最近の概算によれば宇宙の陽子の数はわずか 2^{200} のオーダーである*2 ことを注意しておこう．しかし数学的および理論的な観点からは，これは万能な括弧検査器を持つこととはまったく違うことである．もちろん，これは有限と無限の違いであり，学問的な問題を議論するときにはこれは重要である．われわれは 2 つの任意の大きさの数を加え合わせる FSM を組み立てることが**できる**が，任意に選んだ括弧列を検査できる FSM を組み立てることは**できない**．ついでに言えば，2 進数の掛け算を行なう FSM の作成を妨げているのは，無限のメモリーの必要性である．

チューリングさんと彼の機械に親しむ前に，有限な数の状態を持つ FSM についてさらに 1, 2 の事柄を述べたい．前の章で考察したことの 1 つは，複雑な論理関数をゲートのような単純で基本的な論理単位から組み立てることのできる限界であった．似たような問題がここでも生じる．他のすべての FSM を組み立てられるような FSM の中核の一揃いがあるのだろうか？　この疑問を調べるために，FSM をどのように組み合わせることができるかを調べる必要がある．

図 3-6 に 2 つの機械 **A, B** を示す．私はこの 2 つをフィードバックなどで風変わりなつなぎ方で結合した．ちらっと見たときに，何が起

＊2　太陽の中の陽子数は 2^{190} のオーダーである．現在，地球から見える宇宙の陽子数は 2^{250} のオーダーである．[訳者]

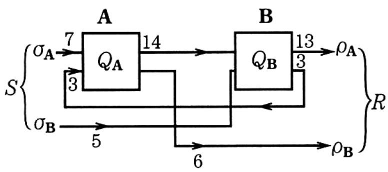
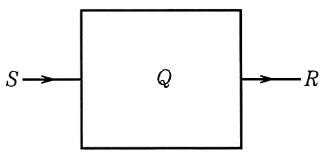


図 3-6　合成 FSM

こっているのかがわからなくても心配はいらない！　この図が何を表しているかを述べよう．一般的な FSM では，入力刺激は任意の 2 進数で，出力もそうである．刺激信号の入力が逐次的であろうと，(たとえば，多くのオン/オフの線による)並列であろうと，この信号を 2 つの組に分割できる．**A** への刺激が 10 ビットだとしよう．これを，たとえば，7 ビットと 3 ビットの刺激に分割する．次が手の込んだ部分である．7 ビットの入力は外部から導線 σ_A を通って入り，3 ビットの入力は**機械 B の応答**から取ってくる．**B** の応答も分割され，たとえば，16 ビットの応答があるとすると，このうちの 3 つを **A** へ送り，その他の 13 を出力とする．我慢して聞いてほしい！　**A** の応答はどうなるだろうか？　これもやはり分割する．たとえばこれを 20 ビットとする．このうち 14 (この数はまったく任意である)を **B** への入力として選び，残りの 6 は **B** を迂回して出力とする．**B** への入力の残りは，それが何であっても，導線 σ_B を通って外部から入る．たとえば σ_B が 5 ビットを伝達するとしよう．

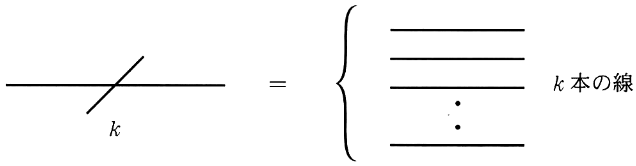
このたわごとの要点は，この複合システムが 1 つの有限状態機械となっているということである．



ここで入力刺激は導線 σ_A と σ_B の 2 進数の入力を組み合わせたもので，出力信号は **A** と **B** からの部分的な応答を組み合わせたものである．明らかに，この機械には 7+5=12 ビットの入力刺激と，13+6=19 ビットの応答がある．この複合装置が何をするかは **A** と **B** の諸性質によって決まる．この結合された機械の内部の状態数は **A** と **B** の状態数の積になるのがもっともらしく思われるが，フィードバックによって影響を受ける範囲や導線を走り回る情報について注意しなければならない．私が示したかったのは，未接続の導線を適切に結んで，小さな FSM からより大きな FSM がどうやって組み立てられるかということであった．たとえばフィードバックを入れ忘れるなど，違う接続にすれば何が起こるかを君たちは知りたいだろう．もし入力と出力の全ビット数への制約をできるだけ少なくしたければ，フィードバックが不可欠なことがわかるだろう．たとえば，出力を入力に直接つなぐことによって，2 つの機械を結びつけることは，刺激/応答の全体の数を固定するだけでなく，複合機械を構成している FSM に入力と出力を整合させるよう要求する．

「基本的な FSM の中核の一揃いを使って任意の FSM を組み立て

ることができるだろうか？」という質問にもどろう．解答はイエスであることがわかる．それら自体が有限状態機械とみなされるおなじみの AND と NOT が中核の FSM で，これらを使って他のすべての FSM を組み立てることができる．どのようにするのかをおおざっぱに示そう．まず新しい表記法が少し必要である．k 本の信号伝送導線の組を，1本の線に短い斜線をつけ，その横に数字 k を書いて表そう．



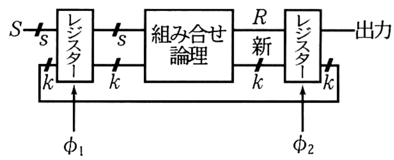


図 3-7　一般的な FSM

この便利な表記法を使って，一般の有限状態機械の説明図を描くことができる(図 3-7)．このかなり複雑に見える装置の処理はまったく簡単である．これは2つのレジスター(第2章でクロックを持つフリップ-フロップから組み立てたようなもの)，およびある論理演算を実行するブラックボックスから構成されている．最初のレジスターへの入力は，FSM への刺激 S と機械の現在の状態 Q の2組である．われわれの設計の拠り所になるのは，内部状態に2進数で名前を付けられるという事実である．この場合，刺激は s ビットで，s 本の導線で入力され，他方，状態は k ビットで，k 本の導線で入力される．(したがって，FSM には 2^k 個までの内部状態がある)．後述するタイミングに従って，このレジスターはこれらの2つの入力信号を論理装置に渡す．ここに秘訣がある．与えられた状態にある FSM は与えられた刺激に応えて応答を作り出し，(おそらく)新しい状態になる．われわれの現在の記述法を使うと，ブラックボックスは2つの2進数を入力として受け取り，2つの2進数を出力として作り出すということに相当する．2つの出力の1つは応答を表し，もう1つは新しい状態を表す．新しい状態の情報は最初のレジスターにフィードバックされ，機械に次の刺激への準備をさせる．そこで FSM が動作するのを保証することは，各入力に対して正しい出力を与える論理装置を組み立てる問題にすぎなくなるが，このことは AND と NOT を正しく組み合わせる問題にすぎないことをわれわれは知っている．

タイミングについて急いで話そう．すでに議論したように，回路設計の実用性は，論理装置の入力と出力をクロックで制御しなければならないことを意味する．信号伝達時間が有限なので，信号の到着時刻にいろいろな遅延が生じるのを許容しなければならない．われわれの FSM も例外ではなく，いつものように，構成部品であるレジスターに2つのクロックを連結しなければならない．この場合の動作方法は標準的な論理回路と本質的に同じである．最初のレジスターはクロック ϕ_1 で，2番目のレジスターは ϕ_2 で動作しており，1つのクロック

がオンになっているときには他のクロックがオフになるように調整する．このためには，ϕ_2=NOT ϕ_1 として両方のクロックを標準クロックに接続し，各々の信号がオンになっている時間が導線上の信号が落ち着く時間に比べて十分な長さ以上であるようにすればよい．重要なのは ϕ_1 が**オン**のとき ϕ_2 が**オフ**であることを保証し，1番目のレジスターが初期状態の情報を処理している間に2番目のレジスターが状態変化についての情報を1番目レジスターに送るのを防ぐことである．

問題 3-4　チューリング機械を考える前に，君たちが考えたくなるような魅力的なFSMの問題を紹介しよう．これは「葬列儀杖(ぎじょう)隊」の問題と呼ばれている．私が「兵士」と呼んでいる任意の数の同一の有限状態機械が一列に並んでいる．たとえば N 個のFSMが並んでいるとしよう．この列の一方の端にはもう1つのFSMである「将軍」がいる．この将軍が「発砲」と叫ぶ．ここで，難問なのは，すべての兵士にできる限り短い時間で，同時に発砲させることであり，次のような制限がある．まず，時間は単位時間ずつ経過していくこと，次に，時刻 $T+1$ での各FSMの状態はすぐ隣の兵士の時刻 T での状態のみによって決まること，3つ目は，考え出した方法が兵士の数 N に無関係でなければならないことである．最初は，どのFSMも静かである．次に，将軍が「発砲」というパルスを出し，これが彼のすぐ隣にいる兵士への入力として作用する．この兵士はある仕方で反応し，新しい状態になり，これが彼の隣の兵士に影響を与え，というように列に沿って影響を及ぼしていく．列の兵士全員はある仕方で相互作用し，…．そしてある時点で彼らは同期して「発砲」を表すパルスを出す．（ついでに言えば，将軍は動作を開始させた後は自ら進んで何かをすることはしない．）

これを実行するいくつかの方法があり，将軍が命令してから兵士が発砲するまでの時間は，普通，$3N$ と $8N$ の間にあることがわかる．すべての必要な情報が行き渡るのに十分な時間が無いため，兵士たちが $T=2N-2$ 以前に発砲することはできないことが証明できる．この最小時間の場合の解は実際に見つけられている．しかし，これは非常に難しいので，解こうと意欲的になりすぎるべきではない．しかし，これは魅力的な問題であり，私はしばしば飛行機の中でこの問題を解こうとして時間をつぶしている．私はまだ解いていない．

3-4　チューリング機械

とうとうチューリング機械について話す時が来た．チューリングの着想は，自分自身あるいは他の数学者を機械と見なすことであった．有限状態機械が頭の中にあって，彼が書くことのできる紙がいくらでもあるような機械としてである．チューリング機械とFSMを区別するものは無制限の紙，つまり実際上は無限のメモリーである．括弧列

検査や掛け算のようないくつかの問題は有限状態機械では実行できないことを思い出してほしい．実行できない理由は，FSMは定義によって無限のメモリー容量が無いからである．この制限はチューリング機械には当てはまらない．そのような機械に付いている紙の量が無限に**ある**と言っているのではない．すべての段階で紙は有限**である**が，紙をもっと欲しいときにはいつでも紙の山を追加できるのである．したがって，われわれは「無制限」という言葉を使う．

チューリング機械の記述法にはいろいろあるが，われわれは恐らく最もありふれた描写を採用しよう．有限な数の内部状態があり，テープの上を動く小さな機械を心に描こう．われわれは紙をテープという形で使うことにした．テープはセルに区切られていて，それぞれのセルには記号が入っている．機械の動作は単純で，FSMに似ている．すなわち，機械はある状態から出発し，セルの内容を見る．その状態とセルの内容に応じて，この機械はセルの内容を消して新しい内容を書き込むこともあるし，セルの内容をそのままにしておくこともある(行動の一様性を確保するために，これを内容を消して再び書きもどすとみなす)．いずれにしても，この機械は次に1セルだけ左または右に移動し，新しい内部状態に変わる．チューリング機械は図3-8のように見える．

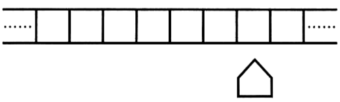


図3-8　チューリング機械

チューリング機械はFSMによく似ていることがわかる．FSMのようにこの機械には内部状態がある．左または右に移動するときにセルの内容を読むことは刺激に似ていて，その内容を上書きすることは応答に似ている．機械が一度に一区画分だけ移動するという制限は絶対に必要な制限ではない．この制限はこの機械をより単純なものにしているにすぎないが，これはわれわれが望んでいるところである．チューリング機械にとって必須の1つの特徴は機械が左と右の両方に移動できることである．一方向だけにしか移動できないチューリング機械は単なる有限状態機械にすぎず，限界もそのままであることが示せる(しかし君たちはもっと慣れるまで待ちたいと思うかもしれない)．

さて，まずテープの有限の部分だけに何かが書かれていることを強調しよう．テープのこの領域の両側は，空白になっている．最初にわれわれは機械に，出発場所と出発時刻Tを告げる．その後の，たとえば時刻$T+1$(チューリング機械はFSMのようにパルス化された時間で処理を行なう)での機械の動作は3つの関数によって規定される．それぞれの関数は時刻Tでの状態Q_iと読み出したばかりの記号S_iに依存する．これらの3つの関数は新しい状態Q_j，書き込む記号S_j，そしてこの後の移動方向Dである．次のように書ける．

$$
\begin{aligned}
Q_j &= F(Q_i, S_i) \\
S_j &= G(Q_i, S_i) \qquad (3.3) \\
D &= D(Q_i, S_i)
\end{aligned}
$$

この一覧表はFSMの仕様書に似ているが，余計な関数Dがある．この機械はこれらの関数によって完全に記述されるが，君たちはこれらの関数を「5つ組(quintuple)」の巨大な(そして有限な)検索表と見なすことができる．「5つ組」とは，われわれが定義した時刻Tにおける2つの関数(Q_iとS_i)と時刻$T+1$における3つの関数(Q_j, S_jおよびD)の合計5つの関数の組の気取った呼び名である．君たちが次に行なうことは，あるデータをテープに書いてそれを機械に読ませることによって機械にそのデータを入力し，機械にどこから出発するのかを告げ，あとは機械自身にどんどん仕事をさせるだけである．このもくろみは，君たちがひまなときに熟読できるように，機械がテープのある場所に計算結果を印刷して機械の作業が終わることである．機械が停止するために，君たちは機械にいつ停止するのかを命令せねばならないことに注意してほしい．これはささいなことのように思われるが，後でわかるように，「停止する」という事柄には，計算に関するいくつかの，非常に重要で，そして非常に深遠な問題が潜んでいる．

チューリング機械のいくつかの具体例を示す前に，われわれがなぜこの機械を考察しようとしているのかを君たちに思い出してもらおう．ある問題を実行するための有効な手順を見つけることと，その問題を解けるチューリング機械を見つけることは同等であると私は前に述べた．ところで，すべてのチューリング機械，すなわちすべての「5つ組」を列挙したとすると，その中には他のどんなチューリング機械ができることでもすべてできるという非常に特殊な種類の**万能チューリング機械**(UTM)が存在する！　これがわかれば，前に述べた同等性によって見通しが開けるだろう．具体的にいうと，あるUTMはより単純なチューリング機械の問題解決活動をまねる模倣機になっている．(私が“the” UTMではなく“a” UTMと言うのは，すべてのUTMが計算上は同等であるが，多くの異なる手法で組み立てられるからである)．「5つ組」の表によって定義されているチューリング機械があって，ある特定の入力データの組が与えられると，ある特定の出力を計算すると仮定する．この機械の処理をまねる1つのUTMは，このチューリング機械の記述，すなわち「5つ組」の表，および入力データの両方をこのUTMに入力することによって得られる．両方の入力は，あらゆるチューリング機械へのデータ入力の場合と同じように，UTMが理解する言語でUTMのテープに書き込むことによって行なう．われわれはUTMに各々がどこから始まってどこで終わるのかも教える[*3]．そこでUTMの内部プログラムはこの情報を受け取って元のチューリング機械の動作をまねる．最後にこの機械は，計算の結果，つまり，元のチューリング機械の出力を吐き出す．UTMの印象的な

*3　UTMのテープのうち，模倣している機械についての情報を含んでいる部分は普通は「疑似テープ」と呼ばれている．[RPF]

ところは，われわれが行なわねばならないことが，この機械に「5つ組」の表とある初期データを与えることだけだということである．すなわち他のどの機械をまねるにしても，そのUTM自体を定義する「5つ組」の組はそのままでよい．どの機械をまねる場合でも，そのUTMの「5つ組」を変更する必要はない[*4]．なぜそのような機械がわれわれにとって重要なのかというと，もし自分**自身**をまねるUTMを作ろうとすると，どのようなチューリング機械も，したがってどのような数学者も解けないいくつかの問題があることを発見することになるからである！

次に実際のチューリング機械のいくつかを調べてみよう．最初の機械は最も簡単なものの1つで，われわれがすでに調べた有限状態機械のパリティ機械に関係がある．この機械に2進数列を入力して，数列の中の1の箇数が奇数か偶数かを知らせるようにしたい．図示すると図3-9のようになる．われわれは，図に示されているように，入力データの2進数列をテープに書き込むことから始める．テープの各セルには1つの数字が入っている．機械の「テープヘッド」は，この数字列の最左端の最初の数字のところで静止しており，われわれは機械が状態 Q_0 にあると定義する．数字列の左側には0だけがあり，右側にも0だけがある．われわれはこれらの0とこの数字列を「終わり(end)」を意味する文字 E で分離し，機械がこれらの0を数字列の一部と思わないようにする．

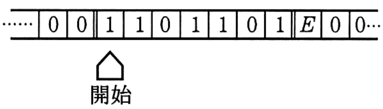


図 3-9 パリティ計数器の入力テープ

われわれがすぐ後で「5つ組」に翻訳しようとしている，この機械の処理は以下のようである．機械の状態はこの数字列のパリティを知らせる．この機械は状態 Q_0 から出発する．この機械はまだ1つも1に出会っていないので，状態 Q_0 のパリティは偶パリティである．もしこの機械が0に出会えば，状態 Q_0 のまま1つ右に移動する．0に行き当たってもパリティは変わらないので状態は変化しない．しかし，1に行き当たれば，機械はこれを消し，0に置き換え，1つ右に移動し，状態 Q_1 になる．次に0に行き当たったら，以前のように，状態 Q_1 のまま1スペース右に移動する．もし1に行き当たったら，これを消し，その場所に0を入れ，右に移動するが，今回は状態 Q_0 に逆もどりする．君たちはこれで何が起こっているかを理解したはずである．1に出会ったら状態を変え，0の列をあとに残していく．もしこの機械がこの数字列の最後の数字を処理した後で状態 Q_0 にあれば，この数字列は偶パリティを持ち，状態 Q_1 にあれば，奇パリティを持つ．この機械はどのようにしてわれわれにパリティを知らせるのだろうか？　それは簡単である．もし機械が E を読めば何を行なうかを機械に教える規則を入れておけばよい．もし機械が状態 Q_0 にあって，E を読み込めば，E を消して「0」を書く．これは偶パリティであることを意味する．状態 Q_1 にあれば，E に「1」を上書きして，奇パ

*4 われわれは後でUTMを実際に組み立てるつもりである．[RPF]

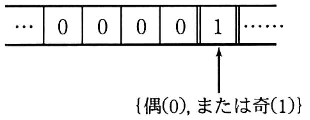


図 3-10　パリティ計数器の出力テープ

リティであることを示す．いずれの場合にも，機械はここで「停止(halt)」を意味する新しい状態 H になる．機械はもはや右や左に動く必要はない．われわれがテープを調べると，ヘッドの真上の数字が問題の答である．この最終状況は図 3-10 に示したようになる．この機械の「5 つ組」は簡単に書き出せる(表 3-2)．

表 3-2　パリティ計数器の「5 つ組」

現在の状態	読む	新しい状態	書く	移動の向き
Q_0	0	Q_0	0	R（右）
Q_0	1	Q_1	0	R
Q_1	0	Q_1	0	R
Q_1	1	Q_0	0	R
Q_0	E	H（停止）	0	——
Q_1	E	H	1	——

さて，この装置はかなり頭が悪い．われわれはすでに有限状態機械を使ってパリティの問題が解けることを知っている．(われわれのチューリング機械が 1 方向だけに移動したことに注意すること！)．FSM では作れないことがわかっている括弧検査器をチューリングさんの創造物を使って組み立てて，その優位性を示そう．しかし，その前に，「5 つ組」の表を苦労して見なくても，機械の動作を容易に理解できるように，新しい図式を導入しよう．

この図式の考え方は FSM で採用されたものと当然のことながら似ている．事実，2 つの図式のただ 1 つの実質的な相違点は，セルに上書きした後のヘッドの移動の向きを何とかして含まねばならないことと，開始と停止の条件を組み込まねばならないことである．他のすべての点ではこの図式は FSM の図式に似ている．われわれのパリティ計数器を記述する図 3-11 を見てほしい．この図は，同じ仕事をする FSM の図 3-5 と本質的に同じである．FSM で刺激があるところには，TM(チューリング機械)ではセルの内容がある．刺激とセルの内容は，いずれも線と円の接点に書かれている．FSM が応答を吐き出したところでは，TM はセルの内容を上書きする．応答と上書き内容は，いずれも状態を結ぶ矢印の上に書く．FSM と TM の状態を表す記号は円の中に書かれている．大きな違いは，まず機械がどこから開始するのかを知らねばならないことであり，これは図に示されているように外部からの矢印を付け加えることによって行なう．そして機械がいつ停止するのかを示さねばならないが，このために機械が E を読めるように別の矢印を各状態に付け加えている．これらの矢印はすべて「停止」H で終わっている．さらに工夫して，われわれは各処理をした後の機械の移動の向きも記述せねばならない．移動の向きが内部状態**だけで**決まり，読み込む記号には無関係な機械が，テープの記号も移動の向きに影響を与えるもっと一般的な機械に比べて，計算

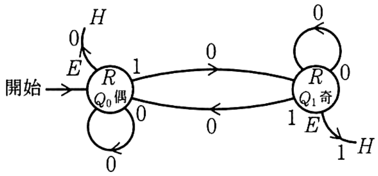


図 3-11　パリティ計数器のチューリング機械

の実行能力の点で基本的には劣らないことがわかっている．そこで，私は右または左への移動が内部状態のみによって決まる機械に話を限定したい．こうすると図式の問題を簡単に解決できる．すなわち，状態円の中に L か R を書けばよい．この例の場合には，両方の状態とも右への移動と結びついている．

君たちがチューリング機械の基本的な仕組と記法に親しむことが重要なので，私はこのかなり頭の悪いパリティの機械について少し長く話しつづけてきた．次にもっと興味深い問題である括弧検査器を組み立てる問題を調べてみよう．この問題は有限状態機械よりもチューリング機械が優れていることを示す具体例である．各セルに括弧が書かれているテープをこのチューリング機械に与えよう(図 3-12)．この括弧列の両端には記号 E が記されている．これは括弧列を表示する明らかに最も簡単な方法である．どのようにすればこの機械に括弧列の妥当性を調べさせられるのだろうか？　1つの方法は次のようなものである．まず言葉でこの方法を説明し，それから状態図などを使って議論する．この機械は括弧列の最左端から出発する．この機械は右括弧に出会うまですべての左括弧を通り抜けて移動する．そしてこの右括弧に X，または「(」や「)」や E 以外の，君たちが選んだ任意の記号を上書きし，1 区画だけ左に移動する．このときこの機械は，左括弧の上にある．これにも X を上書きする．これで一対の括弧を相殺した．X についての重要な性質は，この機械が X を気にしない，すなわち機械には X 記号が見えないことである．このようにして一対を相殺した後，この機械は再び右に移動し，右括弧に行き当たるまで X と左括弧を通り抜ける．右括弧に行き当ると，またこの括弧に X を上書きする．君たちが理解したように，このようにしてこの機械は規則正しく括弧の対を相殺する．この機械のヘッドはそのうちに左右のどちらかの E に行き当たり，決定的瞬間がやってくる．このときこの機械は，2 つの E の間のテープに X だけが含まれているかあるいはつり合っていない括弧も含まれているかを調べねばならない．もし前者ならば，この括弧列は正しく，この機械はこのことをわれわれに知らせるために(たとえば) 1 をどこかに書き込む．もし後者ならば，この機械は 0 を書き込み，この括弧列が間違っていることを知らせる．もちろん，書き込んだ後に，この機械は停止を命じられる．

···· E(())()(()))(()E ····

図 3-12　括弧検査器の入力テープ

君たちがこの処理法を考えてみれば，この非常に簡単な処理法がすべての括弧列を，その長さにかかわりなく検査できることがわかる．この機械の機能は図 3-13 の状態図(ミンスキー[1967][*5]による)に示されている．この状態図が FSM の状態図とどのように異なっているかを注意深く観察しなさい．われわれは開始と終了の命令と，移動の

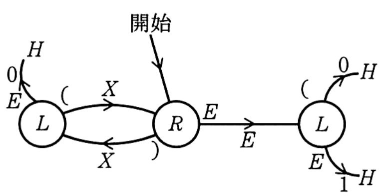


図 3-13　括弧検査器の状態図

*5　Marvin L. Minsky, Computation : Finite and Infinite Machines, Prentice-Hall (1967).〔訳者〕

向きの標識を取り入れなければならない．また，この機械には，パリティ計数器の場合とは異なり，2 つの別々な左移動の状態が必要である．

さて，君たちは基本的な考え方をいくらか理解したので，いくつかのチューリング機械を試しに設計してみたいだろう．君たちが考えるために，いくつかの例題を示す．

問題 3-5　1 進数の乗算器を設計しなさい．「1 進数」は基数 1 で書かれた数であり，2 進数よりもさらに原始的である．1 進数には，数字の 1 だけがあり，数 N は N 個の 1 の数字列として書かれる．たとえば，1＝1，2 (基数 10)＝11(基数 1)，3＝111，4＝1111，など．君たちに任意の 2 つの 1 進数を互いに掛け合わせるチューリング機械を設計してもらいたい．次に示す入力数字列を使って始めなさい．

$$\underbrace{}\;00\cdots0E\underbrace{1111\cdots1}_{m}B\underbrace{1111\cdots11}_{n}E0\cdots00$$

この数字列は掛け合わされる数の m と n を符号化し，2 つの数を記号 B で分離している．目標は答えの mn を表すテープの状態で止まることである．このテープは次のようなものになるだろう．

$$\cdots00E\underbrace{0000\cdots0}_{m}B\underbrace{XXX\cdots X}_{n}E\underbrace{YYYY\cdots Y}_{mn}00\cdots$$

ここで，Y は $0, 1, X, E$，および B のどれとも異なる記号である．このテープの構造は，この問題を解く 1 つの方法への強力なヒントである！

問題 3-6　われわれは以前に 2 進数の加算器を議論した．ここで君たちに 2 つの 2 進数を加え合わせるチューリング機械を設計してもらいたい．しかし(問題を簡単にするために) 2 つの数のビット数が同じ場合だけでよい．記号 A, B, C によって区切られた 2 つの数の領域を持つ数 m と n に対する次のような初期テープを使って始めてよい．

$$\cdots00A\underbrace{1101\cdot\cdot1}_{m}B\underbrace{1001\cdot\cdot0}_{n}C000\cdots$$

この機械の開始場所，処理方法，最終出力の形式，出力場所などについては君たちに任せる．

問題 3-7　もし君たちがこれらの問題がやさしすぎると思うなら，次の問題はずっと難しい．2 進数を掛け合わせるチューリング機械を設計しなさい！

問題 3-8 最後の問題はすばらしい問題である．1進数から2進数への変換器を設計しなさい．つまり，1進数を表す1の列を入力すれば，この数を2進数に変換して出力する機械を設計しなさい．この問題を解く秘訣は除数と余りの計算にある．われわれが2進表記のnビットの数$N=N_n N_{n-1} \cdots N_1 N_0$というとき，これが何を意味しているのかを考えよう．これは定義により，

$$N = N_n \cdot 2^n + N_{n-1} \cdot 2^{n-1} + \cdots + N_1 \cdot 2 + N_0$$

である．われわれは1進数で書かれたN，すなわちN個の1の列から出発して，2進数の数字である係数N_iを見つけたい．いちばん右側の数字N_0は，

$$N = 2 \cdot X + N_0$$

なので，Nを2で割ったときの余りである．この式のXを求めるのはやさしい．N_1を見つけるには，N_0を除いて

$$X = 2 \cdot Y + N_1$$

であることを使う．すなわちXを2で割った余りがN_1である．われわれは，2で割って余りを求めるというこの作業を，2進数の結果が得られるまでつづける．Nはnビットの数なので，定義によってN_nは1**でなければならない**ことに注意すること．

1進数の形で数Nが与えられた場合に，1を2つ1組の対にしていき何が残るかを調べていくと，上の処理をまねられる．具体例を考えよう．10進数の9，すなわち1進数の111111111を使う．1を2つ1組にしていくと，

(11)(11)(11)(11)1

となる．明らかにこれは2で割っているようなものである．右側に孤立した数字があるが，これはN_0が1であることを示している．N_1を見つけるには，右側の1を削除し，残りの数字列の対をさらに対にまとめる．

(11 11)(11 11)

今度は余りがないのでN_1は0である．同様にして，N_2は0になることがわかる．これでわれわれはすべての対および対の対をさらに対にまとめた．残されたのは，この数の左にN_3として1を付けることであり，111111111(1進数)＝1001(2進数)が得られる．

チューリング機械を使ってこの手順を実現するのは君たちにまかせよう．君たちは数字を2つ1組にし，対としての印をつけ，余りを調べねばならない．そこでまた始めにもどって対を2つ1組にしていくなどの作業を行なう．対に印をつけるには数字列の左端から出発して右に行き，1つおきの数字にぶつかるたびにこれを記号Xで置き換えていくのがおそらく最も良い方法だろう．この機械がもう一度この数字列を通っていくときには，この機械はXを無視し，別の1つおきの1に再びぶつかる．この改良された方法は，うまくいく！ この

方法の詳細を考えるのは君たちにまかせよう．この機械の動作を開始させ，変換を実行させ，出力を書かせ，そして停止させねばならないことを忘れないように．

3-5　さらにチューリング機械について

さて，計算の異なった側面にかかわるかなり複雑なチューリング機械を調べよう．私はこの講義の初期に，コンピューターは計算する機械というよりは書類を動かす機械であることを指摘した．そこで算術ではなく，ファイル処理の機能をもつチューリング機械が組み立てられるかどうかを調べることには魅力がある．そのような機能で最も原始的なのは，ファイルの中の情報を調べることであり，これからわれわれが調べようとしているのはこの機械である．われわれは，最初にファイルシステムの中のファイルの場所をつきとめ，つぎにその内容を読み，最後にこの内容をわれわれに中継するような機械が欲しい*6．

われわれは，次のようなチューリングの「ファイルシステム」，すなわちテープ(図3-14)を使い，これをわれわれの機械に入力する．

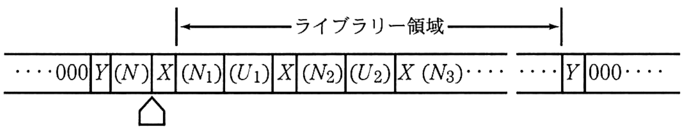


図3-14　場所を突き止める機械への入力テープ

この図は少し構造がわかりやすいように描いてある．記号Xはここではいろいろな登録ファイルを分離する役割を果している．一対のXの間には1つのファイルが登録されている．各登録ファイルは名前(すなわち，アドレス)「N_i」と内容「U_i」から構成されていて，両方とも2進数列で，いつものようにテープの1区画に1数字とする．われわれは機械に読ませたいファイルの名前「N」をテープの左端に付け，テープの左端を記号Yで表示する．この記号の左側には一連の0があり，テープの右端もまた同じようになっている．この機械はわれわれが見つけたいファイルの名前Nの右側のX印のところから開始することになっている．

この機械が直面する最初の課題は，正しいファイルの場所をつきとめることである．機械は左から右へ動きながら一覧表の中の各ファイル名と目標の名前とを系統的に比較することによって，正しいファイルを見つけるまで，この場所をつきとめる作業を行なう．これをどのように行なうのだろうか？　理解しやすいように，図3-15のようなファイルを記録したテープがあるとする．

便利なように，名前の数字列とデータの数字列の両方を同じ長さ，3ビットにしている．われわれは，目標ファイルと呼んでいるファイ

---

*6　われわれの議論は，ミンスキー[1967]の議論を念入りにたどっている．[RPF]
（ミンスキー[1967]の書名については，p.53の脚注5を参照．[訳者]）

$\cdots 000\ Y\ 101\ X\ 001\ 011\ X\ 101\ 110\ X\ 111\ 000\ Y\ 000 \cdots$

N 目標ファイル名　N_1　U_1　N_2　U_2

図 3-15　ファイルテープの例

ル(101)の内容を読みたい．さて，これを行なうための最良の方法は次のようであると思うかもしれない．目標となる可能性があるファイルの名前それぞれに，チューリング機械の別々の状態に割り当てる．これは最大で 8 つの状態になる．この機械は，目標ファイル名で指示された初期状態 101 から開始し，左側から一番目のファイルへ行き，その名前を調べる．一致すればそれで万事終了である．そうでなければ，その右側の次のファイルへ行き，それを調べる．このようにして，この機械は正しいアドレスに行き当たるまで滑らかに左から右へ移動する．しかし，このような機械のかかえる問題はこの機械がたった 8 つの状態だけしか持っていないことであり，3 ビットのファイルシステムに対してだけ有効であるにすぎないことである．したがって，この機械をすべてのファイルシステムに普遍的に使うことはできない．われわれはどのような長さのファイルも処理できる 1 つの機械が欲しい．どのような長さのファイルも処理できる機械は，ファイル名と目標ファイル名を表す 2 つの数字列の間を苦労しながら往復して，数字を 1 桁ずつ比べていく．不一致の桁が見つかった場合には次のファイル名のところへ行く．2 つの数字列が完全に一致した場合には(たとえば)出発点にもどる．テープのすでに検査した部分を見失わないように，この機械は後で無視する記号を数字に上書きするという，括弧検出器で行なったおなじみの芸当を行なう．0 と 1 に異なる記号，たとえば，A と B をそれぞれ割り当てることによって，どれが 0 であり 1 であったのかがわかるようにできる．もしわれわれが次の日にやって来てこのファイルをまた使いたいと思えば，このファイルが異なるアルファベットで書かれていることに気づきさえすれば，このファイルを使うことができる．そのとき，この新しい記号を 0 や 1 で上書きすることによって完全に元のファイルを再現できる．

場所をつきとめるチューリング機械に対するミンスキーの解を図 3-16 に示す．

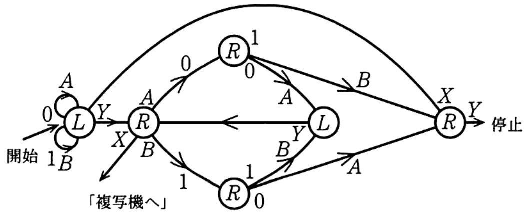


図 3-16　場所を突き止める機械

この図式には，「複写機へ」という，固定されていない端がある．これは機械が正しいファイル名を突き止めて，次に何をしようかと思っている段階を表している．ファイルの情報をどのようにしてテープ上の特定の場所に複写させるのかについてはすぐ後で示す．

しばらくの間はこの場所を突き止める機械の話をつづけ，この機械がどのように動作するかをもっと詳しく調べよう．ヘッドは目標の数字の右側にある最初の X から出発する．円の中の命令(L)が示すように，この機械は左へ向かい，目標の数字の中の0と1をそれぞれ A と B に変える．これは少し奇妙に見えるかもしれないが，後でわかるように，これには目的がある．この機械はやがて Y に行き当たる．そこで新しい状態に入り，この図式から明らかなように，右に動き始める．この機械はまず，書き込んだばかりの A か B のどちらかに出会う．機械はこの記号を元の数字の0か1で上書きし(これは**確かに**奇妙に見えるが，意味がある！)，再び右へ移動する．ここでこの機械は，A か B ではなく，0か1だけを認識する2つの状態のうちの1つに入る．もしこの機械が A または B に行き当たったなら，これを無視し，右へ移動しつづける．言い換えると，この機械はある意味では数字列の最初の数字を書き留めたあとで，書きかえられた目標の数字列の残りの部分を右に通り抜けていく．これが目標の数字列の中の0と1を A と B で上書きした理由である．この機械は出会った X も通り抜けて調べるべき最初のファイル名のところへ進んでいく．

さて，きわめて重大な一連の処理の場面に来た．この機械は1または0のどちらかに行き当たる．このときどのように反応するかは，機械がどのような予備知識を与えられていたか，すなわち目標の最初の数字を読んだ結果としての状態によって決まる．2つの可能性がある．まず，この機械が行き当たった数字が目標の最初の数字と異なる場合は，ファイル名は初めから一致せず，この機械はこの数字を決められたとおり上書きし，そして，ファイルの最後を示す次の X に行き当たるまで右に移動する．それからこの機械は左へ移動し始め，ファイルの残りの内容を A と B で上書きしていく．この機械は最左端の X も通り抜け，目標ファイル名(A と B はこの機械には見えない)を通って勢いよく進み，最初の数字を A または B に変えて Y に行き当たる．これが全過程が再開するきっかけの合図になる．このとき初めてこの機械は次のファイル名のところへ行く．遅かれ早かれ，目標の最初の数字と調べたファイル名の最初の数字とが一致するが，これが考えねばならない第2番目の可能性である．

一致した場合は，この機械はその一致した数字に上書きして，Y に出会うまで左にもどるという状態になる．それから機械は右に前進し，目標の2番目の記号を正しい数字で上書きし，ファイルのところ

へ移動する．この機械は2番目の桁が一致しているかどうかを調べ，そしてこれをつづける．この機械の図式を勉強すれば，上のテープが最終的には図3-17のテープに変換されることが納得できるだろう．

…00 Y 101 X AAB ABB X BAB 110 X 111 000 Y 00…

N △　　　　$U(N)$

図3-17　場所を突き止める機械からの出力テープ

ヘッドが出発点にもどったこと，そしてその活動の効果で，望みのファイル名(しかしファイルの内容ではない)の最初から最後までの間のすべての0と1がAとBに変わったことに注意すること．(われわれがファイル名を間違ってタイプしたというような理由で，目標のファイル名が見つけられないという重要な可能性がある．この場合にはこの機械のヘッドは，図が示しているように，一番右のYに到着し，この時点でヘッドは「停止」命令を受ける．)

前に述べたように，この図には，複写機への「固定されていない線」がある．われわれはファイルを持っているので，このファイルで何をすればよいのかを知りたい！　チューリング機械の精神に忠実に，われわれはこのファイルをテープの別の部分へゆっくりと手間をかけて複写しようとしている．すなわち，**君たち**がしようとしている．この複写機は図3-18に示されている．

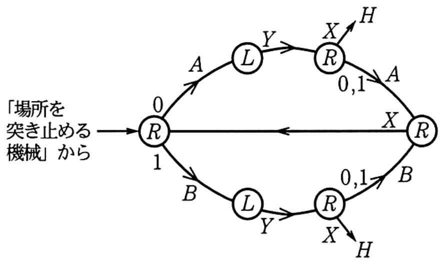


図3-18　複写機

複写機への入力テープは場所をつきとめる機械の出力テープである．この複写機がどのように動作するかを考えるのを楽しんでほしい！この機械のかわいい特徴は，ファイルの内容を，元のテープ上の目標ファイル名を含んでいる区画へコピーすることである．すなわち，目標の数字列は上書きされる．(ファイル名と内容が同じ長さを持つように選んだためにこのようなことができる．)この機械がテープに処理を行なった最終結果は図3-19のテープである．

…Y BBA X AAB ABB X BAB BBA X B11 000 Y…

$U(N)$

図3-19　複写機からの出力テープ

この節を終えるにあたり，チューリングの問題をあと2つ出す．

問題 3-9[*7]　空白のテープから開始し，すべての2進数がテープの上に順に書かれて作業が終わるチューリング機械を作りなさい．これらの2進数は「Y」で分離され，各々の数の終わりを示すYを書いた後ではその数を決して変更しないという制約がある．この機械は書き終わって区切った数を調べることさえしないという付加的な制限を課すこともできる．

問題 3-10　次の形の数字列だけを，そしてその全部を認識する機械を設計しなさい．

$$10110011100011110000\cdots\cdots1^n\,0^n$$

この機械がこのようなテープを「受理した」ということは，この機械が停止して，しかも機械にかけたあとのテープが空白であることによって示される．もっと一般的に，ある任意の数字列が記されているテープをあるチューリング機械にかけた場合，最終的に機械が停止し，テープが空白になっていたら，この数字列はこの機械で「受理可能」であると定義する．われわれはこの概念を有限状態機械を含むように拡張できる．すべてのFSMによって受理される数字列の組だけを正確に受理するチューリング機械を設計せよ．（ヒント：チューリングの「5つ組」を作るためにFSMの関数FとGを使え．）

3-6　万能チューリング機械と停止問題

われわれがチューリング機械を研究している理由に立ちもどろう．前に私は，もしある計算に対する有効な手順があれば，それは同じ計算をするチューリング機械を原理的に見つけられることと等価であると話した．関数を使って話すことが有効である．変数をxとして，この変数の関数を$F(x)$とする．もし2進法，1進法などのうちのある表示でxが書かれているテープを入力すると，最終的にテープに$F(x)$を印刷して停止するチューリング機械$\mathbf{T}_F$を見つけられるとき，関数$F(x)$は**チューリング計算可能**であるという．これまでに見つかっているすべての有効な手順には，同等の計算をするチューリング機械が見つかっている．したがって，一般帰納的関数はチューリング計算可能であり，逆もまた真である．そこで実際的には「チューリング計算可能」は「計算可能」の同義語と考えることができる．

さて，変数xのある値に対してチューリング機械が停止しない場合がありうる．これは不思議な動作だが，起こりうる．x^2のような多くの関数は，機械に入力するすべての値のxに対して，機械が関数の値をテープに書き込んで停止するという意味で「完全」と呼ばれる．こうならない関数は「部分的」と呼ばれる．この場合は関数の操作的定義を次のように変更しなければならない．もし，ある値xに対して機械が停止すれば，その関数の値は$F(x)$であると定義する．

＊7　ある与えられたビット数の2進数に限定した設問と考えられる．［訳者］

もし機械が止まらなければ，その関数の値は0であると**定義する**．これは，関数$x-3$に$x=3$を代入すれば0になるというように，xをFに入れると0になることを意味**しない**．ここで，「0」はチューリング機械が計算をやめないときに，$F(x)$に付ける役立つラベルにすぎない．この再定義された関数は，**すべての**xに対して**ある**数値を割り当てられるという意味で，完全である．

「xがどの値のときに，この機械は動きがとれなくなるのかを，あらかじめ言うことができるだろうか？」という疑問が当然起こる．ある場合には，この答はイエスである．たとえば，機械が2つの状態の間を行ったり来たりして何も実行しないというような，それとわかる無限ループに入ってしまうような場合があるかもしれない．そのようなときには機械は決して止まらないと確実に言うことができる．しかし一般には，特定のxの値がいつ騒ぎを起こすかをあらかじめ言うことはできない！　**別の言い方をすれば，機械$\mathbf{T}_F$が入力xによって停止するかどうかを予言する計算可能な関数を作成することはできない**．なぜそのようになっているのかを調べると，われわれはチューリングさんの小さな機械の能力を高く評価するようになるだろう．

太文字にして目立つようにした前の段落の後ろから2つ目の文章で，私はこれから話すことを予告した．私は，入力xに対して$\mathbf{T}_F$が停止するかどうかをわれわれに知らせる計算可能な関数があるかどうかという疑問を提起した．しかし，そのような関数があれば，その関数は定義によってチューリング機械で記述可能でなければならない．他のチューリング機械についてわれわれに教えるチューリング機械という概念は，われわれがこれから考える**万能**チューリング機械という話題にとって中心的な概念である．

上に記した質問を次のように表すことができる．**D**という名前の機械があるとしよう．**D**は入力として，$\mathbf{T}_F$についての情報と$\mathbf{T}_F$のテープの最初の内容についての情報(すなわちxについての情報)を含んだテープを与えられる．機械**D**は$\mathbf{T}_F$が停止するか否か，イエスかノーか，をわれわれに知らせることを要求されている．重要なのは，**D**が必ずその答を書いて停止しなければならないということである．ここで，**D**からの出力に対して次のように反応する別の機械**Z**を導入しよう．

もし$\mathbf{T}_F$が停止する(D が「イエス」と言う)と，Z は停止しない．

もし$\mathbf{T}_F$が停止しない(D が「ノー」と言う)と，Z は停止する．

そして，この**Z**にそれ自身を処理させて矛盾を見つける！　このような議論を展開しよう．

Dに対する探求を始めるために，まずどのようにして1つのチュー

リング機械に別の機械の動作を理解させるのかを調べる必要がある．われわれは，ある与えられた機械 **T** とそのテープ t の特性を表す必要がある．これにはいくつかの方法があるが，われわれは「5つ組」による記述を選ぶ(表3-3)．

表3-3　チューリング機械の「5つ組」の記述

前		後		
状態	読む	状態	書く	移動の向き
Q	S	Q'	S'	d (=L または R)

われわれはすべての **T** を模倣できる万能な機械を組み立てたい．言い換えると，もしこの機械に **T** および **T** のテープ t についての情報を入力すれば，この万能な機械は，t の上で動作する **T** の結果を出力する．この万能な機械を **U** と呼ぶことにし，その特性を，**T** と同様に「5つ組」を使って表そう．**U** を定義する「5つ組」を$(q, s\,;q', s', d')$と書くことにするが，**U** に模倣させるすべての可能な機械 **T** に対して，この「5つ組」がそのまま使えなければならないことに注意しよう．q, s などは個々の **T** の特質に依存してはならない．われわれがこの機械に課す1つの制限は，テープの記号 S, S', s, s'が2進数でなければならないということである．任意のチューリング機械 **T** には任意の記号が使えるが，考えてみれば，すべての記号は必ず2進数で表すことができ，これらを使って作業できることがわかるはずである(たとえば，8つの記号があれば，各記号は3ビットの2進数列で書き直せる)*8．

U の基本的な動作を記述するのはとても簡単である(ここでもミンスキー[1967]の議論を忠実にたどる)．**U** は各段階で **T** のテープの状態を記録しつづけながら一段階ずつ **T** を模倣する必要がある．**U** は各時点での **T** の状態を書き留めねばならず，そして模倣した **T** のテープを調べることによって，**U** は **T** が任意の段階で読んだ内容を自分自身に伝えることができる．**U** が持っている **T** についての記述を調べることによって，**U** は **T** が次に何をすることになっているのかを知ることができる．ミンスキーはこの過程を，あるチューリング機械が何をするのかを発見するために「5つ組」の表とテープを使って**君たちが実行する**ことに，うまく関連づけた．この万能なチューリング機械 **U** は君たちを遅くしただけのようなものだ！

図3-20に示したテープを **U** に供給しよう．
左側の無限の「疑似テープ」は **U** の作業空間であり，シミュレーションの各段階で **T** のテープがどのようになっているかを **U** が追いつづける場所である．疑似テープを左側だけに無限に選んだことは本質

*8　練習問題として，2^n 個の記号を使って処理するチューリング機械 **T** を0と1で処理する機械 **T**′になるようにするには，どのようにプログラムし直せばよいのかを調べなさい．ヒント：**T** が一度に1つの記号を読み出さねばならない場所で，**T**′は n 個の記号を読み出さねばならない．[RPF]

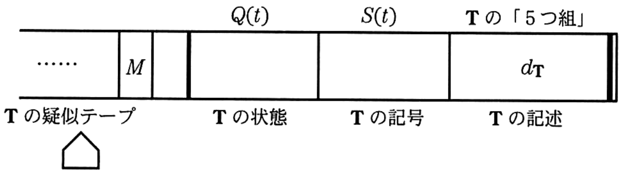


図 3-20　万能チューリング機械への入力テープ

的ではなく，簡単にするためである．標識 M の位置は，**T** のテープヘッドが現在 t 上でどこにあるかを **U** に示す．この作業空間の右側は **T** の状態を含んだテープの部分である．そして，その右は **T** が読んだばかりの記号を含んでいる部分，そして最後に **T** の記述を含む領域がある．d_{T} と記したこの **T** の記述は，**T** の「5つ組」の一連の表から構成されており，2進数列で書かれている(図 3-21)．

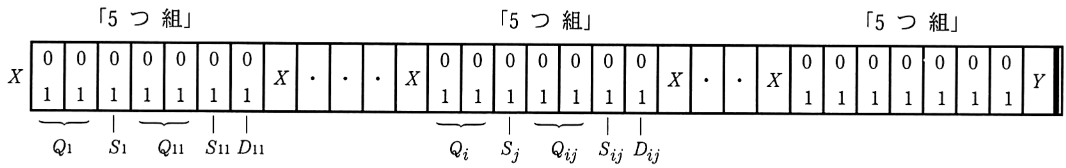


図 3-21　U のテープに対する T の記述 d_{T}

各「5つ組」と次の「5つ組」とは記号 X によって隔離されている．**U** を開始させるためには，**T** の初期状態 Q_0 と **T** が最初に読み込む記号 S_0 を **U** に知らせる必要がある．図 3-22 に示されているように **U** のテープヘッドは最初は一番左の X の上にあるものとしよう．

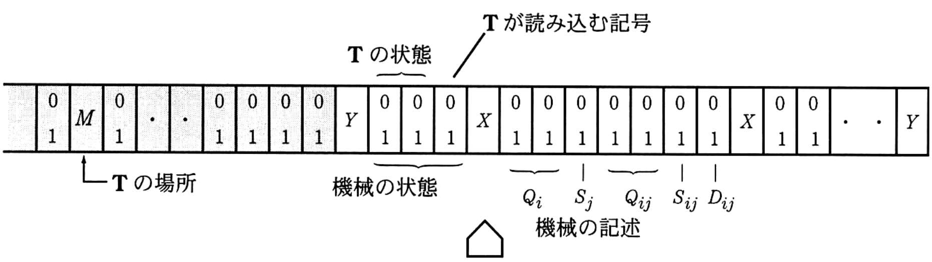


図 3-22　U のテープヘッドの開始位置

U の操作として不可欠なのは，前述した，場所をつきとめる操作と複写操作である．普通の用語を使うと，**U** は次のように仕事をする．まず **U** は，場所をつきとめる機械が行なった通りに，与えられた Q と S に対する **T** の動作を記述するテープの領域の中を見る．この(Q, S)の組は，対応する「5つ組」を含むファイルのファイル名とみなすことができる．場所をつきとめる機械のように，この操作では途中で出会うすべての 0 と 1 を A と B に変える．機械は適切な Q と S の対を見つけると(これらを A や B に変えて)，一番左の X へもどる．

次の段階は複写機に関係している．**U** は 0 や 1 の最初の組に行き当たるまで右に移動する．われわれが与えたこのテープでは，この 0 や 1 の組は Q と S で指定された「5つ組」の残りの 3 つの部分を表

す．この3つは，**T**の新しい状態，この機械が(疑似テープ上の位置Mに)書く記号，およびすぐ後の移動の向きである．この機械は次に，新しいQと新しいSの両方を表すAやBを，テープの中央にある機械の状態領域に複写する．この機械は移動の向きd(AまたはBとして表されているLまたはR)を記憶する．次に機械はMに到達するまで左に向かって進む．Mに到達すると，Mを消して一時的に向きd(AまたはB)を上書きする．次に右に移動し，その途中にあるすべてのAとBを0と1に変えながら(Mの旧位置にあるAまたはBはそのままにしておく)，最後に，この機械はいちばん左のXのすぐ左のところまで移動し，そこにある記号Sを消して(しかしこれを記憶する)そこに特別な記号Vを書き込む(Vが使われるのはこれだけである)．

ここで，この機械は最終局面に入る．この機械はわれわれがMに格納したAまたはBに出会うまで左に移動する．このAまたはBは**T**が次に移動すべき向きdを表している．この機械は記憶していたSをこのAまたはBに上書きし，それから命令dに従って左または右に移動する．機械は，現在いる区画の記号を読み，それを記憶し，そこにMを書き込む．次にこの機械はVに到達するまで右に移動し，このVを記憶していた記号で置き換える．ここでこの一連の操作のすべてが再開する．

この機械が実行したことはTの操作の1サイクルを模倣したことである．この機械はある状態Qと与えられた入力の記号Sで開始した．次に状態を変え，新しい記号を書き，**T**によって指示された次の記号のところへ移動した．**U**は**T**を完全にまねるまでこのようなことをつづける．重要なことは，**U**には停止状態があることである．すなわち，**U**は**T**がいつ停止したかを知り，自分自身を停止させる．

上に示したミンスキーによる**U**の記述では，**U**は8つの記号と23の状態を持つことが要求される．君たちが彼の機械の美しさを評価できるように，それを図3-23に完全に示す．この機械を構成機械に分解することはそれほど難しくないことがわかるはずである．

同じ数の記号を使い，しかし6つの状態しかないUTMを組み立てることは可能である．もし手の込んだことをしたいのなら，必要な状態の数を最小にするために，1つ以上の目的に対して同じ状態を使うという方法がある．たった2つの状態と多くの記号，または2つの記号と多くの状態を持つUTMを組み立てることができる．そのような汎用の機械がそれを記述するために部品をほんのわずかしか必要としないということは驚きである．**何でもできる機械は途方もなく複雑になるべきなのだろうか？**　そうではないというのが驚くべき答である！　UTMをどれほど効率的にすることができるかは面白い問題で

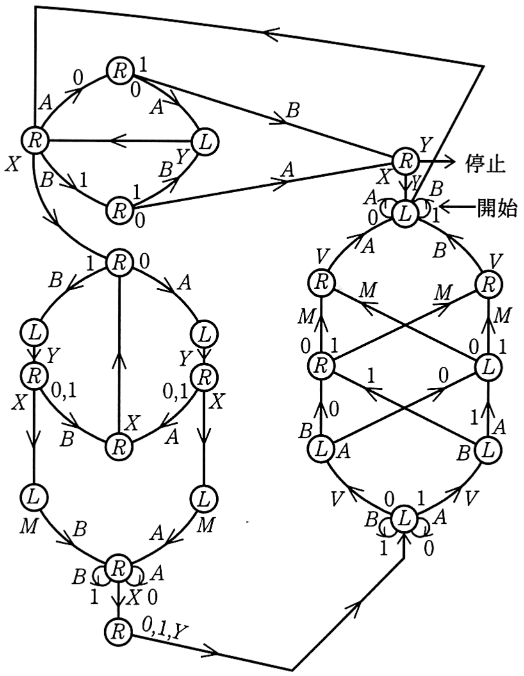


図 3-23 万能チューリング機械

あるが，深い重要性はない．

さて，われわれがUTMの存在を証明することに関心を持ってきた本当の理由に立ちもどろう．われわれは，テープ t を持つチューリング機械 **T** が停止するかどうかを，すべての **T** と t の場合に知らせる機械を組み立てることができるかどうかを問いつづけてきた．われわれはこの問題を万能チューリング機械 **U** の停止問題と言い換えて明確にすることができる．新しい機械 **D** を定義しよう．この機械はテープ t を持つ **T** が停止するかどうかを知らせる，それもどんな機械 **T** とどんなテープ t を対象にしてもできる，という付加的な性質を持つ万能チューリング機械 **U** にすぎない(図 3-24)．言い換えると，**D** は**いつも**答を示して停止する．このような機械は存在するだろうか？答はノーである！　われわれは実際に **D** が不可能な夢であることを示せる．われわれは，**D** が期待されたように実行できないような機械 **T** とテープ t を選ぶことによって，このことを示す．

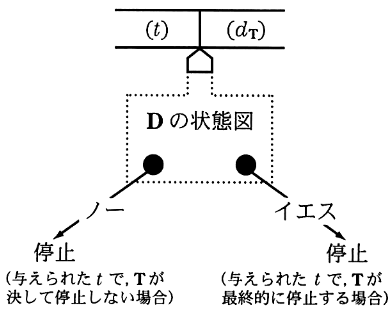


図 3-24 テープ t と d_T を持つ万能機械 D

T と t についての情報としては，d_T という形での **T** の「5つ組」の記述およびテープ t についての情報が万能機械に入力される(図 3-24 参照)．さて，はっきりした理由は無いのだが，もしテープ t に記述 d_T を含めたら何が起こるかを見てみよう．ここで機械 **D** の質を少し高めた別の機械 **E** を導入しよう．この新しい機械は入力として d_T を含んだテープのみを要求する．入力後 **E** は d_T をテープの空白の部分へコピーする．そうすると **E** は，$t=d_T$ と d_T とを含む入力テープ

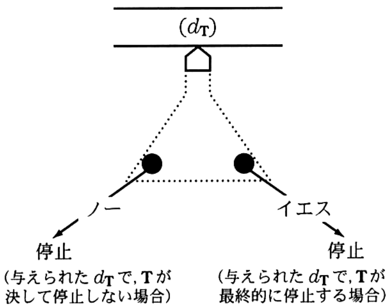


図 3-25　入力テープ d_T を持つ万能機械 E

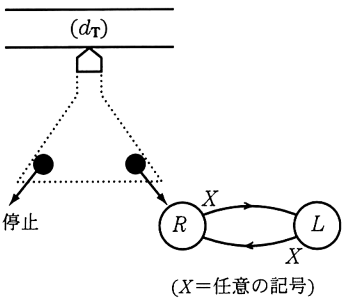


図 3-26　万能機械 Z

を持つ機械 **D** のように動作する．いまや **E** は **D** と同じように動作し，**T** が自身の記述を読んだときに停止する場合は「イエス」と答えて停止する．そうでない場合には，**E** は「ノー」と答える(図 3-25)．どちらの場合にも，**E** は常に停止する．

次に **Z** と呼ぶ **E** の修正版を導入する．この新しい機械 **Z** には，もし **E** が「イエス」と答えた場合に **Z** が停止することを妨げる 2 つの余分な状態がある(図 3-26)．したがって **Z** は，**E** が「イエス」という答を吐き出した場合には停止しない，という性質を持っている．これに対して，**E** が「ノー」を吐き出した場合には **Z** も「ノー」と答えて停止する(すなわちこの場合は **Z**＝**E**)．したがって **Z** に d_T を入力するとき，d_T を適用した **T** が停止**しない**場合は **Z** は停止するが，d_T を適用した **T** が停止**する**場合 **Z** は停止しない．決定的な段階が来た．**Z** に対する記述 d_Z を書き，**T** を **Z** で置き換えよう．そうすると次のような結論が導かれる．

> d_Z を適用した **Z** は，d_Z を適用した **Z** が停止しない場合に停止し，そしてこの場合にのみ停止する．

これは明白な矛盾である！　われわれの議論を振り返ってみると，間違っているのは，**D** が存在するという仮定であることがわかる．**そこで，どのようなチューリング機械によっても解くことのできないいくつかの計算上の問題(たとえば，ある UTM が停止するかどうかの決定)がある．**これがチューリングの主要な結果である．

3-7　計算可能性

多くの計算不可能な関数があるに違いない．計算不可能な関数はいくつあるのだろうか？　数を数えることについて考えると，この問題を見通すことができる．計算可能な実数を考えよう．計算可能な実数とは，この実数の 2 進数展開が，機械が停止しようがしまいがテープに書かれるもの，という意味であるとする．われわれは計算可能な実数よりもずっと多くの実数があることを示せる．その理由は，計算可能な実数は可付番なのに実数は可付番ではないからである．ある集合の要素が正の整数の集合の要素と 1 対 1 に対応づけられる場合，すなわち，それぞれの集合のメンバーにそれぞれ異なる整数を付けられる場合，この集合を「可付番」と呼ぶ．可付番集合の 2 つの例は次に示す偶数と有理数である．

偶数	0	2	4	6	8	10	…	
	0	1	2	3	4	5		
有理数	(1/2)	(1/3	2/3)	(1/4	2/4	3/4)	…	
	1	2	3	4	5	6		

しかし，実数は可付番ではない．この事実の巧妙な証明を以下に示す．実数が可付番だと仮定してみよう．そうすると，実数と整数を 1 つず

つある方法で組にしていくことができる．たとえば次のように，

整数	実数
1	$0.\underline{1}24$
2	$0.0\underline{1}5$
3	$0.53\underline{6}92$
4	$0.800\underline{3}444$
5	$0.3341\underline{0}5011$
6	0.3425…

実数の整数への割当て法は，ここでは奇妙なものを選んだが，任意である．各整数に1つの実数が対応し，すべての実数が組の中に入ればOKである．しかし，そのようにはできない！　その理由を見るために，われわれの表に載りえない実数を見つけよう．上の表の中で，私はいくつかの数字に下線を引いた．最初の数の小数第1位の数字，2番目の数の小数第2位の数字，3番目の数の小数第3位の数字などである．われわれはこれらの数字を使って新しい数を定義する．われわれは，この数の小数第 n 位の数字が表の n 番目の数字とは**違う**ということだけを要求する．実数，

0.22741…

は，どこまでもつづくが，まさにそのような数である．われわれは下線を引いた数字のそれぞれに1を加えてこの数を得た．（この作り方をつじつまの合ったものにするために「9＋1＝0」という規則を含めることができるし，あるいは新しい実数を作るためにまったく別の作り方を使うことができる．）われわれは何を達成したのだろうか？作り方によって，この数は上の対応表の m 番目の数とは小数点第 m 位の数字が異なり，これはすべての m，すなわち，すべての整数に対して正しい．したがって，われわれの表に現れることのできない実数を見つけた．そこで，いわゆる「対角化」によって，われわれは実数が可付番でないことを示した(上の下線を引いた数字の全部を通る直線が「対角」線であるのにちなんで対角化と呼んだ)．

しかし，チューリング機械は可付番**である**．この事実を理解するために，ある機械 **T** のテープの記述 $d_{\mathbf{T}}$ を考えよう．もし各「5つ組」の記載の間の空白を無視すると，この記述をこの機械に固有な2進記号列として考えることができる．この2進数は，整数集合の1つの要素をこの機械の番号としてつけるのに役立つ．他方，n 番目のチューリング機械が停止する場合は関数 $f(n)$ を1と定義し，停止しない場合は0とすると，停止問題でわかったように，明らかにこの関数は計算可能ではない．この他にも多くの例がある．

有効な手順の話題にもどって少しコメントしよう．われわれは有効な手順を，物事を計算できるようにする計算手順(アルゴリズム)として描こうとしたが，現実には多くのそのような手順はほとんど実用性

がない．実行するのに，たとえば，大量のテープや，他のある資源をぜいたくに使用することが必要になるだろう．ある手順を完了するには宇宙の年齢ほどの時間がかかるかもしれないが，それでも技術的には「有効」である．実際には，われわれはただ有効なだけでなく効率的な手順が欲しい．もちろん，「効率的」という言葉を，正確に定義することは容易ではなく，そのためにわれわれは澄み切っていて曖昧なところの無い論理学の世界を離れ，どちらかというと濁っていて漠然とした，あるいは刺激的で面白い現実の世界に入ることになる．現実の世界をどう見るかは君たちの見方次第である．顔の認識のような「人工知能」の分野の多くの問題は効率的ではない有効な手順に関係している．そしてある場合には，それらは非常に有効ですらない！

ときには有効な手順が厳密には必要でないことがある．たとえば，君たちがある質問をし，私は確実に解答することができなくても，$1-10^{-20}$の確率で正しく解答することができる場合がある．君たちはこれほどに良い正解率にきわめて満足かもしれない．この場合の不確実性については特に悪いということはない．このような場合のわかり切っていて面白くない例として，ある数xが別の数yで割り切れるかどうかという問題がある．この問題に対して，私は単に「ノー」ということができる．もしyが十分に大きい数であれば，私が正解である確率の方が高い――正確には，ランダムに選んだ数があるyで割り切れる見込みはy回のうち1回である．ここの基本的な原理は君たちは自分が証明できることよりもずっと多くのことを知りうるということである！　不幸なことに，君たちは実際に知っていることよりもずっと多くのことを知っていると思ってしまうこともある．したがって証明はしばしば必要なのである．

関連したもっと興味深い問題は，与えられた数nが素数かどうかという問題である．この問題の1つの有効な手順は$n^{1/2}$までのすべての素数を持ってきて，nを割り切るものがあるかどうかを探すことである．もし無ければ，nは素数である．この方法は小さなnに対してはうまい方法であり，これでよい．しかし大きなnのときにはこの方法は非実用的になる．もっとよい検査法は確率的検査である．これにはフェルマーの有名な定理の1つ

$$a^p = a \bmod p \tag{3.4}$$

を使う．この式の意味は，任意の数aと素数p（ただし，$a<p$）に対して，a^pをpで割れば，余りとしてaが得られるということである．たとえば，

$$3^5 = 243 = (48\times5)+3$$

となる．この方法の背後にある考えは，大きな値のaを取って，$a \bmod p$を計算するということである．大きな数pに対してはpが素数ではない可能性が高く，pで割った余りとしてとても多くの可能

性があるので，それが a に等しくない確率も高い．(実際の確率を計算するのは簡単ではないが，考え方はわかる．) しかし，もし p が莫大な数，たとえば 10^{200} のオーダーの数であれば，どのようにして a^p を計算すればよいのだろうか？　実際にはこの数は必要でなく，われわれはただ p で割った余りだけが必要なのである．なぜそうなるのかという理由については君たちの練習問題として残しておこう！(一般的な場合を気にすることはない．9 桁の p について考えなさい．)

別の似た問題は素因数分解である．私は君たちに数字 m を与え，これが 2 つの素数の積 $m=pq$ であることを教える．君たちは p と q を見つけなければならない．まだこの問題には有効な手順(「効率のよい手順」ではないかと思われる［訳者］)が無い．そこで，実際に暗号化システムの基礎になっている．この数学的問題の一般的な解法をわれわれが知らないという事実を，メッセージの暗号化法に仕立てることは可能である．ある賢い男が(いまのところ 72 桁の m まで研究されている)この問題を解いた瞬間にこの暗号は使用できなくなり，われわれは別の方法を見つけた方がよくなる．

計算可能性の話題を離れる前に，「文法」という関連した話題のいくつかに言及しておきたい．言語学の場合のように，数学においても文法は基本的には言語の要素を結合するための規則の集合である．異なる点といえば，言語が(算術や代数のような)数学的言語であるということだけである．これらの規則を誤って適用する可能性がある．次のような文を考えなさい．

$$(a+b)c \qquad a+b(c$$

算術の文脈では，最初のものだけが意味がある．2 番目のものは，括弧の位置が間違っており，また無意味に置いてあるので意味が無い．計算における興味深い一般的な問題の 1 つは，数学的な表現(または他の表現)の文法的な正しさを検査する機械を作れるかどうかという問題である．われわれはすでに括弧検査器という 1 つの例を調べた．この機械は「)」と「(」に関する非常に単純な文法を検査し，ただ 1 つの文法規則は括弧列がつり合うということであった．しかしチューリング機械では検査機を作れたが，有限状態機械(FSM)では作れなかったことを思い出してほしい．ところで FSM が検査できる一群の文法がある．たとえば 1 の列 111111… で，正しい列は，たとえば，偶数の数字を持っていなければならない場合である．しかし，この型の機械の能力は限られている．われわれは文法の型をその解析に必要な機械に関連づける表を実際に作成できる(表 3-4)．チューリング機械はとても作りやすいので，この美しい理論の全部を跳び越えていかなければならないのは残念である．それにもかかわらず，(言語の解釈を含む)コンパイラーの設計においてそのような理論を使うことはきわめて重要なので，これについてさらに学べば有益なことがわかる

表 3-4　一般的な文法と機械による実現

言　語	記　述	例	必要な機械
有限枚挙可能	受理可能表現の表	ab, abc	メモリ(表の参照)
正規言語	$*$, $\vee$, $\wedge$, (　)で組み立てられる正規表現	ab^*c, * 印は任意回の繰り返しで，無い場合(0 回)も含む．$a(b\vee d)^*c$	有限状態機械(定理)
文脈自由	帰納を許す生成規則によって生成される言語	a^nb^n($n\neq m$ であるような a^nb^m ではない)	中間的な機械：プッシュダウンオートマトン．1つの「スタック」が内部にある．「スタック」はバネの上に紙を積み重ねたようなもので，一番上の 1 枚だけを出し入れできる．
一般帰納的	計算可能関数	$a^nb^mc^q$	チューリング

だろう．

1921 年に，当時大学院生であったポストが発見した興味深い問題で計算可能性の考察を終えることにする．2 進数列，たとえば 10010 を考えてほしい．この数字列は任意である．この数字列に次のような規則を適用して楽しんでほしい．まず最初の 3 つの数字を読みなさい．もし最初の数字が 0 ならば，3 つを全部消して 00 を数字列の最後に加えなさい．もし最初の数字が 1 ならば，3 つを全部消して 1101 を数字列の最後に加えなさい．そこでこの数字列は

10010
… 101101
… 1011101
……

となる．問題は，この過程は永久につづくのだろうか，終わるのだろうか，または周期的につづくのだろうかというものである．最近私が聞いた話によると，これまで検査された数字列はすべて終わるかループに入るかだったが，一般的にそのようにならねばならないということは証明されていない．ある数字列 g を入力すると，数字列の最初の数字 g_i によって決まる結果 $h(g)$ を出力する機械，いわゆる「ポストの機械」は万能機械として動作すること，およびチューリング機械ができることは何でもできることが示されたため，この問題は興味深い問題である！

第 4 章
符号化と情報理論

この章では計算の抽象的な考察から離れて，コンピューター構造のもっと具体的で現実的な問題を扱う．私は，構成部品の**非信頼性**に起因する機械の限界について調べたい．典型的な機械は何百万という論理ゲートや他のこまごまとした物から組み立てられており，もしこれらの部品にある種の誤動作を起こす傾向があれば，機械の行なう処理に重大な影響を及ぼすだろう．

部品は主に 2 通りの方法でわれわれを困らせる．まず最初に，部品には欠陥があるかもしれない．欠陥は製造中に生じることがあり，明らかにきわめて重大である．たとえば，シリコンからメモリーチップを作るとき，欠陥はいたるところに生じうる．たとえば，材料の中に小さなごみがあったところや製造機が誤って動作したところなどである．そして非常に小さな欠陥でもメモリー素子をだめにする．もし君の記憶装置が全部のセルが動作しなければ実用にならないのであれば，たった 1 つの小さな誤りでも非常に高価な損失になる．この問題を解決する 1 つのうまい方法は，欠陥を見つけ，恐らく以後その部分を使えないように封じることによって，欠陥を避けて動作するシステムを設計することである．部品の物理的な構造については後で調べるつもりである．

いま注目したいのは，素子がわれわれを困らせる第 2 の場合である．これは原子の乱雑なブラウン運動や回路の単なる不規則なノイズなどのために，時たま素子が**ランダムに**誤動作する場合である．そのような故障はどれも，部品を一時的または永久に動作しなくしてしまう可能性がある．さて，ある特定の素子がそのような故障をする確率は 100 万分の 1 かもしれないが，われわれの機械に素子が何十億もあれば，何千という欠陥が，いたるところでどの時刻にも生じるだろう．最も初期のノイマン型計算機が運転されていた頃，この機械は故障率の高い(1000 分の 1 のオーダー)真空管とリレーから組み立てられていた．100 万の部品があるとすると，常にそのうち 1000 個が故障していると予測されるわけで，非信頼性の問題は深刻であった！　いまではトランジスターを使ったずっとよいシステムが開発されてきたので，われわれが組み立てるほとんどすべての機械の故障率は下がりつづけてきた．事実，最近ではこの問題が非常に深刻だと考えられるこ

とはなくなった．しかし，ますます多くの部品を持つコンピューターを製造し，これをますます速く動作させるにつれ，そして特に部品をますます小型化するにつれ，故障がまた問題になるかもしれない．現在のトランジスターにはざっと10^{11}個の原子が含まれているが，もしこの数を減らし，たとえば1000個の原子でスイッチング素子を組み立てようとすれば，雑音やランダムな故障の影響が非常に大きくなる．将来の発展を見据えて，非信頼性の問題をある程度詳細に調べることが賢明である．その上これは興味深い課題であり，それだけで研究するに十分な理由になるだろう．

4-1　計算と通信理論

非信頼性の議論を，非信頼性が最も問題になるコンピューターの部分であるメモリー(記憶)装置の考察から始めよう．たとえば，われわれがあるデータをある場所に長い間格納しているとき，ある時点でシステムが誤りをおかし，ある場所の1ビットを変更した，たとえばある1を0に変えた，と仮定しよう．この種の誤りは機械の他の場所，たとえばCPUの中でも起こりうるが，トランジスターおよび素子の数がはるかに多いメモリーの中でほどは多く起こらないと考えられる．この状況を調べるため，私は工学の別の分野である**通信理論**との類似性を利用するつもりである．通信システムでは，一方の端の送信器でビットの一群を送り出し，他端の受信器でそれらを受け取る．これはただメッセージを送るだけである．この過程で，誤りは知らぬ間に入り込む可能性がある．雑音は伝送中のメッセージに影響を及ぼすかもしれないし，受信が不調になるかもしれないし，機械を故障させるかもしれない．これらのすべては，われわれが受信するメッセージが送られたものとは違いうることを意味している．これがいわゆる「雑音がある場合の通信問題」である．さてこの問題は，空間ではなく時間を通してメッセージを送ると考えられるメモリーにおける問題とは正確には同じ状況ではないが，その類似性はわかるだろう．われわれはメモリーに何かを格納し，後でそれを読み返す．そして，その間に格納された「メッセージ」は雑音を受ける．格納されたメモリーの信頼性と送信されたメッセージの信頼性については，重要な実際的相違がある．たとえば，メモリーの内容を連続的に調べることは原理上は可能だが，NASAが木星探査機に無線通信を送る場合には，伝搬中にその内容を調べる方法は無い．そうではあるが，類似性が十分にあるので通信理論を考察する価値があるものにしている．誤りを訂正する前の不可欠な段階である，誤りの検出方法についての考察から話を始めよう．

4-2　誤りの検出と符号の訂正

これから，私は通信の言葉を使うつもりであり，その言葉を機械のメモリーシステムに関連づけることは，特別の場合以外は，読者にゆ

だねたい．送信されたメッセージがあると仮定しよう．これは，記号*1の列で，われわれはそれを受信しようとしている．言うまでもなく，われわれは受信したメッセージの信頼性を知りたいが，これが誤りの検出という最初の問題をもたらす．受信したメッセージが正しいかどうかを知る方法があるだろうか？　この場合，われわれが対象にするのはメッセージだけであり，送信者に確認の電話をすることは目的に反する！　われわれがメッセージを確認できるように，メッセージ自体に検査を組み込む何らかの方法があるだろうか？　この答は，すぐ後でわかるように，イエスである．

4-2-1　パリティ検査

まずメッセージに誤りが生じる確率は非常に小さいと仮定し，実際上2つ以上の誤りが生じる心配は無いとしよう．さらに，われわれは個々のビットの誤りだけを考え，たとえば隣接した，あるいは関連したいくつかのビットに広がる誤り(たとえばディスク上の傷によって起こる「エラーバースト」など)は考えないことにする．1つの記号が誤りになる可能性は1万分の1だとし，メッセージの長さは記号10個だと仮定しよう．このメッセージに誤りがある可能性は約1000分の1である．しかし2つの誤りが起こる確率は100万分の1のオーダーなので，これを無視できると考える．2つの誤り(二重誤り)はきわめて稀にしか起こらないので心配いらないと仮定して，1つの誤り(単一誤り)を検出することにだけ専念しよう．

パリティ検査と呼ばれる，単一誤りを調べるための非常に簡単な方法がある．次のような10ビットのメッセージを送ると仮定しよう．

1101011001

われわれが行なうのは，この数字列の終わりにこの数字列のパリティを示すビットを付け加えることである．数字列のパリティとは，数字列が含む「1」という数字の個数，または同じことだが数字の和を2で割った余りである．言い換えれば，もし元のメッセージに1が奇数個あれば付け加えられる数字は1であり，偶数個あればこのメッセージには0が付く．これはメッセージの**符号化**の一例である．すなわち，誤り訂正の可能性を(そして，もちろん正しいという保証の根拠も)与えるために，メッセージの基本構造をある方法で修正する一例である．われわれはメッセージを受け取ると，パリティビットを調べる．もし数字1の個数が間違っていたら明らかに誤ったメッセージを受け取ったことになる．この簡単な検査によって奇数個の誤りはすべて検出できるが，偶数個の誤りの場合は検出できない(とはいっても，われわれは2個以上の誤りの確率は0だとしたので，これらの誤りは起こらないと仮定している)*2．

この処理法には，これから述べる2つの主要な欠点がある．第1の

---

*1　これから後では，これらの記号を2進数の数字1と0に限定する．[RPF]

欠点として，パリティビットに誤りが生じるかもしれない！　この場合には，メッセージ自体が誤りであると思ってしまう考え違いをすることになる．明らかに，メッセージが長ければ長いほど，パリティビット自体に誤りが起こる可能性は小さくなる*3．第2の欠点は，うまくいった場合でもこの検査は誤りが存在するか否かを示すだけで，誤りがどこにあるかは示さない．誤りを見つけるためにできるのはメッセージをもう一度送ることだけである．われわれの場合，コンピューターを使っているので，単に機械を再起動して最初の状態にもどせばよい．この方法の他の小さな欠点は，これが通信をいくらか非効率にしていることである．この場合，10ビットのメッセージを伝えるために11ビットを送らねばならないので，10パーセントの非効率になる．

次に考えるべき質問が「誤りの存在を検出するだけでなく，誤りの場所を実際につきとめる方法を構築できるだろうか？」であることは明らかである．答はここでもイエスで，その方法はとても巧妙である．この方法は，われわれが調べたばかりの単純なパリティ検査法を一般化したものである．われわれが行なうのは，メッセージのデータを(たとえば) m 行 n 列の長方形に配列できると考えることである(表4-1)．

表4-1　長方形のデータ配列

	n
	1101 …… 01
	0001 …… 11
m	… … …
	… … …
	1100 …… 01

もちろん，データはこの形で送られるのではない．メッセージは2進数列として送られる．この数字列をあらかじめ定義された規則に従って n 個の記号の m 個の組に分解し，m 個の組を上から下に順に並べる．誤りを検査するためにわれわれが行なうことは，各行の終わりに行パリティを示す数字を，また各列の下に列パリティを示す数字を含めることである．これらのパリティの数字を数字列のメッセージの中に入れるのは難しくない．配列中の誤りは，誤りが現れる行と列の両方でパリティを合わなくするので，われわれが誤りの位置を正確に突き止めることを可能にする．誤りが異なる行と列に起こる限り，メッセージの誤りがいくつあっても，原理的にこの方法で検出できる．

しかし，パリティ検査ビットで起こる誤りには気をつけねばならない．特にやっかいなのは，メッセージの1つの数字と(たとえば)その行パリティの数字が両方とも変わる二重誤りだろう．列パリティが間違っていることから誤りがあったことがわかるが，行パリティの誤りについての確証がないので，列パリティビットの単一誤りであると思いがちである．しかしこの不明確な場合に対しては，この配列の右下にパリティ検査ビットをもう1つ置くことによって保護できる．このビットは全メッセージ(すなわち行の合計と列の合計)のパリティを示

*2　この検査を行なえる単純な機械は，前章で議論したパリティFSMであることに注意すること．[RPF]

*3　単位長のメッセージあたりに，パリティの誤りに起因する誤りが生じる可能性のことだと考えられる．[訳者]

し，これを使ってこのような二重誤りを検出できるが，位置を突き止めることはできない．この二重誤り検出の結果，単一誤りの検出の場合と同様，最初の状態にもどってもう一度メッセージを送ることになるが，それでも改善になっている．

このような符号化法の効率を定量化するために役立つ1つの方法は，**冗長度** R と呼ばれる量の計算である．

$$R = \frac{\text{全体で使われるビット数}}{\text{メッセージのビット数}} \tag{4.1}$$

R が大きくなるにつれて，符号の効率は低下する．$R-1$ という量は普通「超過冗長度」と呼ばれている．最初に述べた単一誤り検出符号の場合には，冗長度は $(n+1)/n$ になる．上の長方形の配列の場合には，$(m-1)(n-1)$ ビットのメッセージを送るために mn ビットを使っている．そこで次のような結果が得られる．

$$R = \frac{mn}{(m-1)(n-1)} \tag{4.2}$$

配列が正方形，すなわち $m=n$ のとき，この量は最小になり，したがってこの符号が最も効率的になる．君たちは「すると，m と n を非常に大きく取るだけで冗長度を下げて1に近付けることができる．列を10ビットではなく1万ビットにしてメッセージを大量に送ろう！」と言いたくなるかもしれない．この場合の問題は，ある確率で各ビットが誤っているので，もし君たちが送るビット数をあまり大きくすると多重誤りが入り込み始めることである．

4-2-2 ハミング符号

今度はパリティ検査にもとづいた，単一誤り訂正(そして二重誤り検出)のもう1つの符号化法を調べたい．これは長方形の方法よりも効率的でしかもはるかに巧妙である．実際には，これは長方形法を高次元へ一般化したものの一種である．メッセージのビット中のいくつかはメッセージ自体によって定義されているが，残りはパリティビットのような符号化記号である．どのようなメッセージに対しても，われわれは「単一誤りの**位置を見つけ出す**だけでなく，それを**訂正する**には検査ビットがいくつ必要だろうか？」と質問できる．この問いに対する1つの賢い答はハミングによって発見された．彼の基本的な考えは次のようなものであった．メッセージに含まれる数字は，互いに重なりのある多くの部分集合に分解される．われわれは各部分集合のパリティ検査を行なう．メッセージに誤りがあると，これらの検査のうちのいくつかあるいは全部が失敗するだろう．われわれはパリティ検査の結果から，明確に定義された規則を使って「シンドローム」と呼ばれる2進数を構成する．もしシンドロームがゼロならば，すべてのパリティ検査に合格したことを意味し，誤りはない．もし0でなければ誤りがあり，さらに**シンドロームの値は誤りの正確な位置を示す**．

たとえば，もしシンドロームが101であれば，これは10進数の5なので，誤りはメッセージの5ビット目にある．もし110010であれば，誤りは50番目のビットにある．この方法の「たね」は「シンドローム」の構成法にある．

検査ビットが何個必要なのかについては直ちに答えることができる．検査ビットがm個になるようにするため，シンドロームの長さがmビットだとしよう．もしシンドロームが0なら誤りが無いことを表すとすれば，最大で(2^m-1)個の，メッセージ上の誤りの位置を符号化できる．しかし，誤りはわれわれが送っている元のメッセージばかりではなく，シンドロームでも起こりうる．したがって，もしnを元のメッセージの長さとすれば，次の関係がある．

$$2^m-1 \geqq n+m \qquad (4.3)$$

すなわち

$$n \leqq 2^m-m-1 \qquad (4.4)$$

たとえば長さ11ビットのメッセージを送りたい場合には，少なくとも4ビットのシンドロームを含まなければならず，全メッセージの長さは15ビットになる．これは特に効率的とは思えない(効率＝11/15すなわち約70%)．しかし，元のメッセージの長さが，たとえば1000ビットの場合，シンドローム($2^{10}=1024$)は10ビットだけでよく，これはかなりの改善になる！

次に，このシンドロームという考え方がどのようなものかを詳しく調べよう．例として，長さ11ビットのメッセージを送る問題をつづけよう．すでに見たように，この場合には4個の検査ビットが必要である．各検査ビットは，全15ビットのメッセージのビットのある部分集合に対するパリティ検査になるだろう．単純なパリティ検査と同様，メッセージ中のいくつかの特定のビット(部分集合)を選び，その全体(部分集合と検査ビット)のパリティを計算し，これが0になるように対応する検査ビットを調整する．もしこの部分集合に1つ誤りがあれば，このパリティ検査は失敗する．ハミング符号の巧妙なところは，各メッセージビットが1つ以上の部分集合に含まれていて1つ以上のパリティ検査に寄与しているが，必ずしも全部のパリティ検査には寄与してはいないことである．どのパリティ検査に失敗し，どれに合格するのかを調べることによって，われわれは誤りの位置を一意的に決めることができる．失敗したパリティ検査には1を，合格したものには0をそれぞれ割り当て，できたビットを2進数として並べる．これがシンドロームである．シンドロームは誤りの位置そのものを示している．並べ方は任意だが，われわれはメッセージ中のパリティ検査ビットを左から右へ読んでシンドロームを作成することにしよう．

さしあたり，パリティ検査ビットがある順番でメッセージ全体の中

に置かれていると仮定しよう．しかし，それがどこなのかについては今のところ気にしない．まず各パリティ検査ビットが受け持つ部分集合を決めよう．このためには，メッセージ中の位置を4桁の2進数表現で表にするとわかりやすい．

1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

一番右端のパリティ検査ビット，つまりシンドロームの右端の数字を調べてみよう．これが0でないとしよう．このとき，2進数表現が1で終わる位置，すなわち1, 3, 5, 7, 9, 11, 13，または15のうちのどれかに誤りがある．これらの位置がわれわれの作る最初の部分集合である．第2の部分集合を得るために，右から2番目の数字を調べよう．これは2, 3, 6, 7, 10, 11, 14および15に対してのみ0ではない．何が起こるのかを注意してほしい．これらのパリティ検査に両方とも失敗した，すなわち各パリティ検査に1を割り当てたとしよう．このとき，誤りは両方の部分集合に共通の位置，すなわち2進数で$ab11$ (a, bは未知)という形になる数の位置にあるに違いない．これは3, 7, 11，または15だけである．つまりわれわれは誤っている可能性のある位置を絞った．これらのどれなのかを知るためには，残りのパリティ検査を行なわねばならない．第3のパリティ検査は位置4-7および12-15についてである．最後のパリティ検査は8から15についてである．これらが両方とも0，すなわち，パリティ検査に合格したとすると，2つの0をこのシンドロームに入れる．そこで4-7, 12-15, 8-15の位置(これらは明らかに重複している)に誤りは無い．しかし3, 7, 11, 15のどこかに誤りがある．後者の4つの中で，先の部分集合に含まれないのは位置3である．誤りがあるのはここに違いない．もちろん，2進数で3は0011であり，これはパリティ検査から計算されたシンドロームである．

このような考え方をもっと具体的に説明するために，例を挙げよう．

11ビットのメッセージ10111011011を送る場合を考える．まず，パリティビットをどこにさし込むのかを決めねばならない．これらがメッセージ中のどこになければならないのかについてはこれまで述べなかったが，実際にはどこにでも入れられる．しかし，ある位置に置く方が他の位置に置くよりも符号化が容易なので，われわれはこのようなもののうちの1つを使う．具体的には，検査ビットを位置1, 2, 4, および8に置く．そこで次のようになる．

符号	a	b	1	c	0	1	1	d	1	0	1	1	0	1	1
位置	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

重要なのは，左から右へ読んだ検査ビットは，シンドロームの逆順になることである．つまり書いたときにシンドロームの最初の左端になる数字は，パリティ検査aではなくdから読み込まれ，最後の右端になる数字はaから読み込まれ，dからではない．これも計算の単純化のためである．

これでa, b, c, dがわかる．ビットaは，奇数位置1, 3, 5, 7, 9, …のパリティである．これは1なので，aは1である．ビットbは位置2, 3, 6, 7, 10, 11, 14および15のパリティの和をとると得られる．これは0である．ビットcは位置4から7と位置12から15のパリティ検査から求められ$c=1$になる．最後に，dは位置8から15のパリティ検査をすることによって得られ$d=1$になる．ところで，この検査ビットの配置がどのようにして素直な符号化に，すなわち$abcd$の計算につながるのかに注意してほしい．もしわれわれが一見もっと簡単な選択をして，$abcd$が数字列の左端の位置1から4になるように選んだら，われわれはa, b, c, dを求めるために1組の連立方程式を解かねばならない．われわれの選択の重要な特徴は，パリティ検査ビットの位置1, 2, 4, 8がそれぞれただ1つの部分集合のみに現れ，a, b, c, dを求める方程式が4つの独立な方程式になるということである．したがって，完全なメッセージは，

101101111011011

である．伝送中に誤りが生じたら何が起こるかを調べよう．次のメッセージ

101101011011011

を受け取ったとする．どこに間違いがあるのだろうか？　奇数位置の数字の和は奇数なので，これは間違っている．そこでシンドロームの右端の数字として1を割り当てる．2番目のパリティ検査の結果も1で失敗になるので，シンドロームは目下，$xy11$である．ここでxとyもこれから決めねばならない．ビットyは次の検査で決まり，これも1なので失敗になる！　シンドロームは目下$x111$である．xを決めるために位置8から15までのパリティを検査すると，これは0，すなわち合格である．したがって，xに0を割り当てて，このシンドロームは0111になる．これは位置7を表す．実際に元のメッセージと

壊れたメッセージを比べると，位置 7 だけが違っていることがわかる．

ハミング符号の興味深い特徴は，メッセージビットと符号ビットが同じ立場にあることである．すなわち符号ビットの誤りは，メッセージビットの誤りと同じようにして突き止められる．二重誤りを検出できるようにこの符号を拡張するのはきわめて簡単である．このメッセージの終わりにもう 1 つ検査ビットを付け，今度は全体のパリティを表すようにすればよい．上に示した(壊れていない)メッセージの場合，パリティは 1 なので右端に 1 を付け，いつものように全体のパリティが 0 になるようにする．さて，メッセージの中に 1 つだけ誤りがある場合は，長さ 15 ビットのメッセージのパリティが変わり，16 番目のビットとつり合わなくなる．しかし二重誤りがある場合は，15 ビットのメッセージのパリティは変わらず，16 番目のビットでは全体が正常に見える．しかし 15 ビットのメッセージのパリティ検査には失敗するので，これが二重誤りを示す．もし 15 ビットは検査に合格するが，全体のパリティが合格しなければ，これは全体の検査ビットに誤りがあることを示す．このような利点のために，11 ビットのメッセージに対して 5 つの検査ビットという，ほぼ 50% の非効率性という対価を支払うことに注意してほしい．しかし前に指摘したように，メッセージの長さが増すにつれて非効率性はかなり低下する．1000 ビットのメッセージに対しては，非効率性はわずか 1% 程度になる．

効率をほんの少し減少させるだけで，メッセージの誤りの影響がどれほど急減するのかを調べることを通じて，ハミング符号のような誤り検出符号の実用性を検討するのは価値のあることである．別々に符号化された，それぞれが約 1000 ビットのいくつかのバッチ(まとまり)でメッセージを送るとしよう．ビットの 1 つが誤りとなる確率が 10^{-6} で，1 つのバッチあたりの誤りの確率が約 10^{-3} になると仮定しよう．われわれはこれらの誤りの確率がランダムで互いに独立であると考える．そうすると，これらのバッチを送るときに起こる多重誤りの確率を取り扱うのにポアッソンの法則が使える．期待される誤りの平均数を m とすると，k 個の誤りが起こる確率は

$$\frac{1}{k!} m^k \exp(-m) \qquad (4.5)$$

である．1 つのバッチあたりの誤りの期待数は，前に述べたように，10^{-3} である．したがって，1 つのバッチに二重誤りが起こる確率は高精度で $(1/2)\cdot 10^{-6}$ になり(指数関数は無視できる)，三重誤りの確率は $(1/6)\cdot 10^{-9}$ になる．さて，誤り検出や訂正が組み込まれていない，いわば保険金を掛けていないと仮定する．この場合に誤りが生じたら，無価値なメッセージを受け取ることになるので，このシステムは機能しない．平均して，このようなことが起こるまでに 1000 個のバッチが送れるだけであり，かなり悲惨である．今度は，単一誤り検出はあ

るが訂正は無い，たとえば単純なパリティ検査だけがあるとする．この場合には，システムが誤りを検出したとき，このシステムは少なくとも停止して私に教え，そして私はこのシステムをどうにかしようとすることができる．誤りはまだ約 1000 バッチあたりに 1 回起こるが，この改善は 1000 個のメッセージビットあたり 1 個のパリティビットという，わずか 0.1% の非効率性という対価で得られたものである．このシステムは，200 万個のバッチごとにおよそ 1 回生じる，二重誤りがある場合にはいつも失敗する．しかし，単一誤り訂正および二重誤り検出ができて非効率性が 1% の装置があると仮定する．このシステムは単一誤りに対しては自分自身で処理する．しかし，約 100 万バッチごとに 1 回の二重誤りが起これば，停止して問題があることを知らせてくれるだけである．このシステムが失敗するのは三重誤りが起こる場合だけであるが，これはおよそ 60 億個のバッチあたり 1 回起こるだけである．1% の投資としては悪い率ではない！

効率と信頼性の問題は，もちろんコンピューター工学において中心的な問題である．当然質問すべき問題は「メッセージの長さはどのくらいにすべきか？」である．長くすればそれだけ符号化の効率に関しては良くなるが，誤りが生じることがより多くなり，もし誤りが起こったなら見つけ出すのにより長い時間がかかる．他方，われわれがメッセージを調べ，受け取っているものが信用できると定期的に感じられるようにするためには，より密に符号化されているが短いメッセージを送らねばならないので，保証に対しては効率を犠牲にする覚悟が必要である．注目に値するこのような場合の例は，ボイジャーシリーズのような太陽系の奥深くへ入る宇宙船との通信の分野にある．宇宙船の費用が何十億ドルもかかり，何百万マイルもの遠距離に無線通信でメッセージを送らねばならず，しかも途中で変わらずに届くように可能な限り確実に送らねばならない場合――木星や土星を調べねばならない時にカメラを太陽に向けたくはない――，効率が低下することを覚悟しなければならない．宇宙船との通信は，「多数決論理決定」と呼ばれているある種の投票技術に頼っている．この方法では元のメッセージの各ビットは奇数回送られる．奇数個のビットの大半は変わらずに，すなわち正しく受信器に到達するだろうというアイディアである．受信器は各ビットのまとまりで多く現れる方のビットを，最良の民主主義流にメッセージビットとして受け取る．コピーを何個送るかは誤りの期待値によって決まる．いずれにせよ，これは通信分野の小さな例にすぎず，私はこれについて詳しく話すつもりはない．計算とあまり多くの関連があるようには思われない(たぶんファイルを 12 回バックアップすることを習慣にする人は例外として)．

4-2-3　メモリーについての余談

ハミング符号化の技術をコンピューターの記憶システムに応用した

面白い方法について手短かに議論しよう．並列処理の出現により，複数の機械が情報をますます速くのみ込んだり吐き出したりするようになったので，情報をますます迅速に書き込んだり取り込んだりすることが必要になってきた．この情報は今までどおりにディスクに格納されているが，個々のディスクはデータの要求された流入と流出を処理するのに十分な速さでは全然ない．その結果，「ギャングディスク(集団ディスク)」システムを使うことが普通になっている．このシステムでは多くのディスクが同時にデータを取り込んだり吐き出したりして負荷を分担している．このシステムは個々のディスクの誤りに明らかに敏感である．たった1つのディスクがだめになっただけで，全ディスクの努力は無駄になる．すべての製造者は完璧なディスク，すなわち誤りの無いディスクを造りたがっているが，悲しいことにこれは不可能である．実際に市販されているどのフロッピーやハードディスクでも，欠陥(たとえば埃，かき傷)がない確率は実質的に0であるというのが正しい*4．君たちが普通これに気づかない理由は，機械が欠陥を見つけ，欠陥を避けて動作するからである．コンピューターがディスク上の悪いセクターをつきとめた場合，普通はこのセクターを封じ，次の良いセクターへ行く．このすべてが非常に速く行なわれるのでわれわれは気がつかない．しかし，あるディスクが並列処理環境で他の多くのディスクといっしょに動いているとき，1つのディスクがある欠陥を処理するためにちょっとの間だけ動かなくなるとすべてがだめになってしまう．

ハミングの方法を使うとこれを救済できる．32枚のディスクがいっしょに動いているとしよう．これらのうちの26枚には情報を入れ，6枚は偽物だとする．われわれは26枚分だけデータを持っているが，このシステムには32本の線が入っている．クロックが時を刻むたびに各ディスクから1ビットが入力される．われわれが行なうことは，システムに各ディスクから1ビットずつ入る32ビットの入力データごとに，(26ビットのメッセージに対して6パリティビットの)ハミングの方法に従ったパリティ検査を実行し，やってくる単一誤りを訂正することである．この種の構成では，2個以上のディスクで同じディスク位置に誤りが生じる二重誤りが起こる確率は非常に大きい．もちろん，これは可能性である．しかし，もしそのようなことが起こるとしても，これらの二重誤りを検出し，われわれが修復するために一時停止するように，システムを改良することができる．ここで話している欠陥は，実際には決してランダムではない．この欠陥は永久的なものとしてディスク上に付いているので，このディスクが動いているときにはいつでも同じ場所に姿を現す．われわれはこのシステムを動かし，欠陥を手直しし，同じ箇所に同時に誤りが生じるすべてのディスクを廃棄することによってこの種の二重誤りを避けることができる．

*4 ディスク製造業者に連絡して製品の欠陥率についていくつかの数値を得ようとしてみなさい．成功するのは容易でないことがわかるだろう！[RPF]

そして新しいディスクを購入する．われわれは二重誤りが見つからなくなるまでこの作業を行なう．このハミング符号法の使用はギャングディスクというアイデアが無駄になるのを救った．

ここに，君たちのためにいくつかの問題がある．

問題 4-1　a 個の要素を持つアルファベットで書かれたメッセージに対するハミング型の符号を考案しなさい(2進数の場合は，$a=2$)．もしハミング符号の記号の数が r で，全メッセージの長さが N ならば(元のメッセージの長さは $N-r$)，次の関係がなければならない．

$$1+(a-1)N \leqq a^r$$

簡単な例について解きなさい．

問題 4-2　これは数学的には面白いがそれほど実際的ではない練習問題である．われわれは**完全な**符号を次のようなものとして定義する．

(a)受信した符号化された各メッセージは，ある意味のメッセージへ復号でき，

そして

(b)もしある特定の個数より誤りが少なければ，この復号されたメッセージは正しい．

符号は e 個の誤りまでを訂正できる(これまでわれわれは $e=1$ だけを考えた)．$e=3$，すなわち三重誤りを訂正できる2進数の記号($a=2$)に対する完全な符号を構成できるか？　ヒント：12ビットのメッセージと11ビットの符号(またはシンドローム)の $N=23$ の場合のみを考えよ．3要素($a=3$)のアルファベットの場合にも解が存在する．この場合，6個のデータビットと5個の符号の $N=11$ を試せ．

4-3　シャノンの定理

この本では多くの基礎的な質問をしてきた．さて別の基礎的な質問をする時がきた．「原理上は，どこまで誤り訂正をできるのだろうか？」「2, 3, 4, 5, 6, 7, … 個などの誤りを訂正する符号は，誤り率が非常に低くなってそれ以上考えても意味がなくなるまで作れるのだろうか？」　失敗する確率の許容限界を 10^{-30} と設定しよう．もしこれが嫌ならば，10^{-100} でもよい．どのような数でもよいが，0はだめである．0にすると後で困る．私は 10^{-30} が良いと思っている．

長さが M_C のメッセージを送信していると仮定しよう．M_C は符号化されたメッセージ全体の長さで，元のデータと符号ビットを含んでいる．いつものように2進数で考える．データメッセージの長さを M と記し，任意の単一ビットが誤りになる確率は q だと仮定しよう．われわれは M_C における単一，二重，三重などの誤りを，より多重の

誤りとなる可能性がわれわれの選んだ数 10^{-30} 未満になるまで，訂正する符号化の仕組みを設計したい．何個の符号ビットが必要になるだろうか？　M_C のうちのどれくらいがメッセージになり，その残りはどのくらいになるだろうか？

シャノンは次のような不等式が M と M_C に対して成り立つことを示した．

$$M/M_C \leqq f(q) = 1-(q \log_2[1/q]+[1-q]\log_2[1/(1-q)]) \tag{4.6}$$

この不等式を導いて，バッチの長さ M_C に制限がなければ，**訂正されずに残る誤りが生じる確率はいくらでも 0 に近づけられる**，とシャノンは言っている．言い換えれば，答は，「はい，われわれは選んだ任意の精度に対して n 重の誤りを訂正する符号を構成できます」である．原理上ただ 1 つの制限は不等式(4.6)である．しかし実際には，バッチの長さを長くして，多くの工夫をすることが必要であろう．シャノンは彼の並はずれて強力な定理によってこれができると述べた．不幸にも，彼は**どのように**するのかを述べなかった．これは別の問題である！

符号化の効率に対するシャノンの上限を例で示すために，いくつかの値の q の場合を表に示す(表 4-2)．q が 0.5 の場合，すなわち，われわれが受け取った任意のビットが誤りである可能性が半々の場合には，どのようなメッセージもまったく届けられないことに注意してほしい．この事実は明らかに意味をなしている．誤りの確率が下がるにつれて，効率の上限は増加する．これはデータビットあたりに必要な符号ビット数は減少することを意味している．しかし，任意の q の値が与えられた場合，シャノンの限界に到達するのは非常に難しい．

表 4-2　シャノンの符号化効率の限界値

q	M/M_C	M_C/M
1/2	0	∞
1/3	0.082	12.2
1/4	0.19	5.3
0.1	0.53	1.9
0.01	0.919	1.09
0.001	0.988	1.012

この定理を実際に証明するのは容易ではない．まず私は君たちにこの定理がどこから来ているのかについて見通しを与える手軽な正当化をしたい．その後で私は幾何学的な方法を使うつもりだが，最後に，別の方法でこの定理を証明する．この方法は物理学に関係し，情報と呼ばれている量の定義を含んでいて，前の方法とは完全に異なっている面白い方法である．しかし，まずこの定理の手軽な正当化をしよう．M_C が非常に大きいと仮定することで話を始めよう．この仮定によってある近似が可能になる．もし単一ビットが誤りになる確率が q ならば，1 バッチあたりに期待される誤りの平均数は，

$$k = qM_C \tag{4.7}$$

である．実際の誤りの数はこれより多いかもしれないし少ないかもしれない．しかしこれはわれわれが期待する平均の誤りの数であり，粗っぽい推定としてはこれで十分である．われわれは，これらの誤りを除去するのにどれくらいの符号化が必要なのかを計算しなければなら

ない．この平均数の誤りが1つのバッチ全体の中に分布する場合の数は，初等的な組合せ理論によって与えられる．

$$\frac{M_C!}{k!(M_C-k)!} \tag{4.8}$$

m 個の符号ビットがあると仮定しよう．この m 個のビットは 2^m 通りの記述ができる．このビット数では少なくとも M ビットのデータメッセージに加えて，誤りの可能な分布の各々について(誤りの検出だけでなく，誤りの**訂正**ができるために)その正確な位置を記述できなければならない．この分布については，$M_C!/k!(M_C-k)!$ 通りの場合があると述べた．(いくつかのビットは余分になりうるので) $m \leqq M_C-M$ であることは明らかである．そこで，次の不等式が得られる．

$$2^{M_C-M} \geqq \frac{M_C!}{k!(M_C-k)!} \tag{4.9}$$

ここで両辺の自然対数をとる．右辺は，大きな n に対するスターリングの公式を使って，近似的に表せる．スターリングの公式は

$$n! = \sqrt{2\pi n}\,n^n e^{-n}\exp[(1/12n)-(1/360n^3)+\cdots] \tag{4.10}$$

なので，

$$\log n! \approx (1/2)\log n + n\log n - n + O(1/n) \tag{4.11}$$

である(ここで，$\log x = \log_e x$)．最後の項は，n が大きくなるにつれて小さくなる(極限では0になる)雑多な項と $\log 2\pi$ のような項を表している．n が大きいときには初項，すなわち $(1/2)\log n$ はその次の2つの項より小さいので，この項は除くことができる．そこで

$$\log n! \approx n\log n - n \tag{4.12}$$

を使うと，不等式の右辺は次のようになる．

$$M_C\log M_C - M_C\log(M_C-k) + k\log(M_C-k) - k\log k \tag{4.13}$$

これは，$k=qM_C$ と代数学を少し使うと，次のようになる．

$$M_C[q\log_2(1/q)+(1-q)\log_2(1/1-q)] \tag{4.14}$$

ここで自然対数を底2へ変換しているが，これは単に両辺に乗数因子を導入するにすぎず，この乗数因子は相殺する．不等式の左辺の2を底とする対数をとり，両辺を M_C で割ると，最終的にシャノンの不等式になる．

この不等式は，ビットの誤り率が q の場合，k 個の誤りを訂正できるようにメッセージ M を符号化したいのであれば，符号化した結果の効率は式(4.6)の限界を越えられないことを示している．もちろん，k は任意ではない．われわれはこの値が誤りの平均数 $k=qM_C$ になるように取った．われわれが答を知りたい質問は，ある数，たとえば k' 個を越える個数の誤りの生じる確率が，確実に 10^{-30} のようなわれわれが選んだ数値になるようにメッセージを符号化できるかということである．シャノンの定理はこれができると言っている．なぜそうなのかを調べるために，この「証明」をもう少し進めてみよう．

メッセージの中に生じうる誤りの数は常に k になるわけではなく，ある範囲および確率内で k になる．事実，誤りの分布は 2 項分布にしたがい，平均値は qM_C $(=k)$，標準偏差は $\sigma=\sqrt{M_C\cdot q(1-q)}$ である．M_C が大きくて q が小さな場合には，この 2 項分布が同じ平均値 qM_C $(=k)$ と標準偏差 $\sigma=\sqrt{M_C\cdot q(1-q)}$ を持つガウス分布(正規分布)で近似できることは確率論の標準的な結果である．さて誤り率がある数 N (たとえば 10^{-30})未満になっていることを要求することは，訂正しなければならない誤りの数が k' 未満になるように要求することと等価である．ここで k' は

$$k' = k+g\sigma \tag{4.15}$$

で，g は N によって決まる有限な数である．ガウス分布の場合には，誤りの数が平均値 k から標準偏差以内に存在する確率は 68%，標準偏差の 2 倍以内には 96%，標準偏差の 3 倍以内には 99% になる．そのため，たとえば，もしこのメッセージに誤りが無いことに 95% の信頼性を持たせたいのであれば，k 個の誤りではなく，

$$k+2\sigma \tag{4.16}$$

個の誤りが訂正できることを要求しなければならない．この場合 $g=2$ である．確率の大きさをどのように選んでも，g の値を見つけることができる．一般に，k から標準偏差の g 倍の範囲外に誤りを見つける確率は

$$\exp(-g^2/2) \tag{4.17}$$

となり，比較的小さい g でも誤りを起こす確率が驚くほど稀なことがわかる．たとえば g が 20 なら，この因子は $\exp(-200)$，すなわち，およそ 10^{-100} になる．われわれの 10^{-30} よりものすごく小さい！ 10^{-30} に対しては，g としておよそ 6 が得られる．

この事実を利用するために，式(4.9)の k を，われわれが訂正したい誤りの新しい個数 k' で置き換えた式を使う．もしそれでもまだ訂正を実行する符号を見つけることができれば，送信中に誤りが生じる確率は，この 10^{-30} 以下，またはどんな数値であってもその値以下であることがわかる．次式，

$$k' = k+g\sigma = k+g\sqrt{k(1-q)} \tag{4.18}$$

を不等式に代入して，$M_C\to\infty$ の極限をとれば，シャノンの結果が前のように出てくることを示すのは読者への練習問題として残そう．

シャノンがわれわれに与えたものは効率の上限である．彼はこの上限に到達する符号化の方法を見つけることができるかどうかについては教えなかった．できるだろうか？　この答はイエスである．私は後でもっと詳しく説明するつもりだが，その技法は基本的には手当たり次第の符号化の仕組みを選んで，M_C をどんどん大きくすることである．これは大変な数学的問題である．もし M_C が十分に大きければ，この符号化の仕組みが何であっても上限の効率に達することができる．

しかし，メッセージの長さは莫大にならねばならないかもしれない．M_Cをどれほど大きくとらねばならないかについてのよい実例は衛星通信にある．地球から木星や土星へメッセージを送るとき，誤り率qが3分の1のオーダーになるのは例外的な場合ではない．これに対する効率の上限は，われわれの表から，8%である．すなわち，各データビットに対して約12個の符号ビットを送らねばならない．しかし，これを行なうためには禁止的に長いM_Cが必要で，あまりに長すぎて実用的でない．実際には，各データビットに対して約**150**個の符号ビットを送るという仕組みが使われている！

4-4　メッセージ空間の幾何学

さて，シャノンの定理を別の角度から調べよう．今度は幾何学を使って「メッセージ空間」という役立つ考え方を導入する．これは主として通信で重要な考え方であり，われわれは計算科学をやっている．しかし私は，この考え方が面白く役立つことがわかった．君たちもそう思うだろう．

メッセージ空間は，簡単に言えばわれわれが送信したいメッセージから構成された空間である．われわれは空間を，各点に座標が付いていて，連続的あるいは離散的(とびとび)な多次元のものとして考えるのに慣れている．メッセージ空間は多次元の離散的空間であり，その点のいくつかあるいは全部はメッセージに対応している．問題を少し具体的にするために，3ビットの2進数符号を考えよう．可能な単語は

000, 001, 010, 011, 100, 101, 110, 111

である．これらは0から7までの2進数である．図4-1に示されているように，これらの数は3次元空間の立方体の頂点の座標と考えることができる．この立方体は3ビットメッセージに対応するメッセージ空間である．この空間を構成する点は立方体の頂点だけであり，図のこれらの点の間の空間や辺などはその構成部分ではない．この空間のすべての点には可能なメッセージが入っているので，この空間はぎっしり詰まっている．もしあるメッセージの1ビットを変えると，別のメッセージになる．無駄は無く，すべてのメッセージには意味がある．4ビットメッセージに一般化するのはやさしい．4ビットメッセージのメッセージ空間は16個の頂点を持つ「超立方体(hypercube)」だが，不幸なことにわれわれの脳はこの空間を視覚化できない！ mビットメッセージの場合は2^m個の頂点をもつm次元空間が必要である．

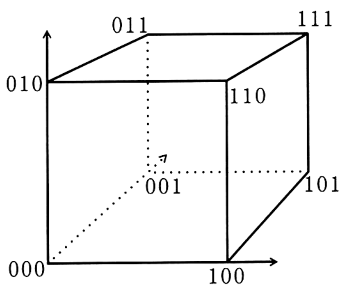


図4-1　単純なメッセージ空間

もし送信中に誤りが生じたら何が起こるだろうか？　この誤りは送ったメッセージのいくつかのビットを変えるが，これはメッセージ空間の別の点への移動に対応する．直観的には，多くの誤りがあればメッセージ空間の中で「遠く」へ移動すると考えてよい．たとえば図

4-1では，(111)は(001)や(100)よりも(000)から「遠く」離れている．そこでこのメッセージ空間にいわゆる「距離関数」を導入する．われわれが使用する距離関数は**ハミング距離**で，2点間のハミング距離は異なるビットの個数として定義される．したがって，111から000へのハミング距離は3で，一方，001から000への距離は1である．この定義によれば，4次元空間での1101から1110への距離は1101から0100への距離と同じになる．距離という考えには意味がある．距離の概念は誤りの議論で役立つからである．明らかに，単一誤りはメッセージ空間の1点からハミング距離が1だけ離れた別の点へ移動させ，二重誤りはハミング距離が2だけ離れた点へ追いやる．誤りの数がeの場合，メッセージ空間の超立方体の各点を中心とする半径eの「誤りの球」を描ける．この球はe個以下の誤りが生じた結果として，球の中心から到着できるメッセージ空間のすべての点を含む球になっている．これは符号化の過程の考察にうまい幾何学的な方法を与えてくれる．

われわれがメッセージMを符号化するときはいつでも，より長いメッセージM_Cに書き換える．M_Cに対するメッセージ空間をMに対する空間と同じように作ることができる．もちろん，M_Cに対する空間はもっと大きく，もっと多くの次元と点がある．明らかに，この空間のすべての点をM空間の点と1対1に関連づけるわけではなく，いくつかの余分な点がある．この余分な点は実際には符号化にとって中心的なものである．e誤り訂正とは，伝送中に最大e個の誤りが発生した場合でも元のメッセージを確実につきとめられるように，符号化された正当なメッセージをM_C空間内に配置することを意味する．この幾何学的な描像では，正当なメッセージはM_Cのメッセージ空間のいくつかの点に対応している．これらの点は誤りによって別の点に移動するが，誤りを訂正するには，正当なメッセージに対応しない点に移動していることがわかったら，正当なメッセージに対応する1つの点に一意に引きもどせるようになっていなければならない．これを保証する1つの方法は，M_Cのメッセージ空間で，どの正当な符号化メッセージに対応する点の間も，ハミング距離が少なくとも

$$d = 2e+1 \tag{4.19}$$

だけ確実に離れているようにすることである．なぜこれでよいのかを示そう．ある正当なメッセージMを送ったときに，e個の誤りが伝送中に生じたとする．受け取ったメッセージM'は，M_C空間で元の位置からe単位だけ離れた点にある．どのようにしてM'からMへもどるのだろうか？　距離が$d=2e+1$以上離れていると要求したので，MはM'に最も近い正当なメッセージである！　他のすべての正当なメッセージは，M'からハミング距離が$e+1$以上離れたところにある．単なる誤り**検出**はもっと安く手に入れられる．つまりこの場合，M_Cで正当な点が互いに$2e$以内にあるようにできる．点が$2e+1$離れて

いるべしという要求は，e 個の誤りを訂正し $2e$ 個の誤りを検出することを可能にしている．

われわれが行なったことを，メッセージ空間 M からメッセージ空間 M_C への写像と描写できる．この写像では M の各要素を，ハミング距離が $2e+1$ 単位以内に別の正当な点が無いような M_C の点へ関連づけている．われわれは M_C 空間を，正当な符号化メッセージ点を中心とする半径が e 単位で内部が詰まった球で作られていると描写できる(図 4-2)．もしこれらの球の 1 つの内部のどこかにあるメッセージを受け取ったら，どの点が元のメッセージに対応しているかが正確にわかる．

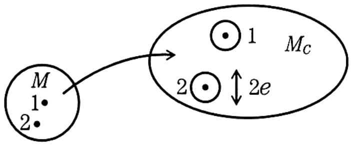
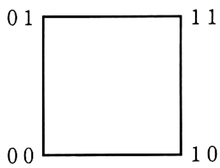


図 4-2　メッセージ空間の写像

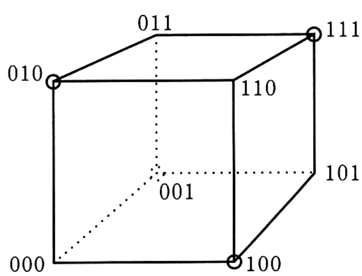
この考えをもう少し具体的にするために，以前に描いた 3 ビットの立方体にもどろう．われわれはこの立方体を，2 ビットのメッセージ

00, 01, 10, 11

から構成された 2 ビットのメッセージ系 M に対して，単一誤りは検出するが訂正はしないパリティ符号に対応するメッセージ空間 M_C と考えることができる．この系は単純な 2 次元の正方形のメッセージ空間を持つ．



パリティ符号は M の各メッセージに余計な 1 桁を付けるので，その結果 M_C のメッセージ空間は 3 次元の立方体になる．正当なメッセージは 000, 011, 101 および 110 である．余分な 4 つの頂点が残っている．



伝送中に生じたすべての誤りにより，これらの余分な頂点のどれかに移り，誤りが生じたことを示すが，それがどこに生じたかはわからない．すべての偽りの頂点は，本物の頂点からハミング距離が 1 のところにあることに注意すること．もしこの系に対して単一誤り訂正をしたいのならば，4 次元の M_C 空間を使わねばならない．

もしこの符号化システムが機能すれば，各メッセージ点がメッセージ空間 M_C のどこかに移動したときに，互いに十分離れているようにできる．時々いくつかの誤りの球同士が重なるようにせざるを得ないが，これは普通は問題ではない*5．次にこの幾何学的な方法を使って，シャノンの定理の別の証明を手短かにしてみよう．元のメッセージの次元と符号化されたメッセージの次元をそれぞれ M, M_C とする．メッセージ列の次元というのは，メッセージ列の長さの気取った呼び名にすぎない．M 空間の点の数は 2^M 個で，M_C 空間の点の数は 2^{M_C} 個である．k 個の誤りを訂正するためには，M 空間の各点を中心とする半径 k の誤りの球を M_C 空間に詰め込む必要がある．これらの球は重なり合ってほしくない．この条件を使うと，M_C 空間の体積と球の体積とを関係づける不等式が得られる．さて，メッセージ空間のような種類の離散空間では，球の体積はその中に含まれる点の数として定義される．M_C 次元空間の場合，半径 k 単位の範囲内に入っていて，1 単位間隔で配置されている点の数は，

$$\frac{M_C!}{k!(M_C-k)!} \tag{4.20}$$

である*6．M_C の体積が各誤り球内の点の数と球の数(すなわち，M の点の数)の積以上でなければならないことに注意すると，不等式(4.9)が得られる．この後の導出をもう一度やり直す必要はない．君たちはどのようにすればこの証明ができるかわかるはずだ．

問題 4-3 メッセージ空間を使って(私が解いた方法で)解ける良い問題がある．もう君たちは，単一誤りを検出するためにメッセージの終わりにある単一のパリティビットを使い慣れている．この技法の 1 つの特徴は，メッセージがどれほど長くても 1 つのチェックビットだけが必要なことである．すなわち M_C に独立である．問題は「M_C に独立な二重誤りを検出する方法も作れるか？」である．われわれは訂正には興味がなく，検出だけに興味がある．われわれは常に同じ数の，有限のビット数を望む．これはわずか 2 ビットでできそうに思われる！ しかしハミング符号では，全体のパリティのチェックビットとシンドロームを使って単一誤り訂正と二重誤り検出ができたが，シンドロームに寄与するチェックビットの数はメッセージの長さに依存したということを思い出してほしい．二重誤りを実際に検出するのは，メッセージの長さに依存する数のチェックビットを使わない限り不可能なことがわかるはずである．

4-5 データ圧縮と情報

すぐにシャノンの定理を他のもう 1 つの方法で考察するが，その前

*5 以前の問題で導入した「完全な」符号は，誤りの球が重なることなくメッセージ空間を「満たす」ものである．もし球の半径が e なら，この球の中の各点はただ 1 つのメッセージ点から e 単位の範囲内にある．[RPF]
*6 式(4.20)は，半径 k 単位の球の表面積と考えられる．[訳者]

に少し横道にそれさせてもらいたい．横道とはまずデータ圧縮のことであり，その背後にある考え方のいくつかを説明したい．英語のような言語を考えよう．英語には26文字があり，コンマ，終止符，空白等を加えると，通信のための約30種類の記号がある．これらの記号(文字)を10文字使って，どれほど多くの事柄を英語で言うことができるだろうか？　君たちは30の10乗と答えるかもしれない．これは間違いである．もし私が次のような列

cpfajrarfw

を書いたら，これは英語ではない．現実の解釈可能な英語には，そのすべては入っていない．つまり，単語として認められるものの数は限られていて，単語の中での文字の順序はランダムではない．もし「T」があれば，次の文字として可能性が大きいのは「H」である．「X」になることはなく，「J」は稀である．なぜだろうか？　文字は一様に使われるのではなく，英語で利用できるメッセージの数は思ったよりもはるかに少ない．

おそらく，というかほとんど間違いなく，君たちの両親は君たちが決して手紙を書こうとしないという事実に故郷で悩んでいるだろう．そこで両親は君たちに返信住所など，何でも書いてある葉書を送る．葉書の裏には多くの四角がある．その四角の1つには「私はカルテクで愉快に楽しく過ごしている．イエス/ノー」と書いてある．次のメッセージには「美しい娘またはハンサムな若者に会った」とかなんとか「イエス/ノー」と書いてある．次のメッセージには「私には送る洗濯物はもうない」とか「何月何日に洗濯物を送る」などと書いてある．かわいそうな両親がしていることは，長い英文を1つのビットに変換することである！　両親は君たちに反省させて手紙を書かせようとしているが，もしそれに失敗しても，手紙を書くよりも1ビットあたりの効率がはるかに良い通信手段を君たちに提供しているのだ．(もちろん，君たちはそれでも葉書を投函しなければならない！）君たちのだれもが行なったことがありそうな，効率を改善する別の例は，長距離電話会社を無料で利用するものである．もしパーティーに出席するつもりなら電話の呼び出し音を前もって決めてある回数だけ鳴らし，もし行かないのなら別の回数だけ鳴らす．あるいは，3回だけ呼び出し音を鳴らしたら，5分以内に車で彼の家に行くので外で待っていてほしいという合図である等．君たちは長距離電話をかけるが，相手は受話器を取らないので，メッセージを送るのに料金がかからない．

関連した疑問は，「英語はどのくらい非効率的か？」ということである．10個の記号を英語のアルファベットをそのまま使って送るべきだろうか？　あるいは他のアルファベットを使って送るべきだろうか？　仮に30個ではなく32個の記号(文字)があるとすれば，各記号(文字)を5ビットの列で表せる．したがって10個の記号(文字)は50

ビットになり，2^{50} 個の可能なメッセージが得られる．もちろん，すでに述べたように，ほとんどのメッセージは意味をなさない．Qが現れるときはいつでも，その後にUがつづくことが期待される．したがって，われわれはUを送る必要がない．もちろん例外としてIraqのようなやっかいな言葉があるが，これはQQを送ることで処理できる．そこでわれわれは言語の構造を利用して，もっと少ない記号で送れるようにできる．このメッセージの詰め込み，すなわち圧縮が，われわれがこれから考察する課題である．

詰め込みという考えを調べるのによい方法は，メッセージの中のある記号を受け取った場合に「その次の記号によってわれわれがどのくらい驚くだろうか？」と質問することである．つまり，Tを受け取った場合，次にIを受け取ったら驚かないが，XやJなら驚くだろう．もしTを選んだら，次にJがくる可能性はきわめて小さい．つまり，英語では，任意の文字を選べるという自由度が無い．そこで，どれだけの自由度があるのかを推測したい．そのための面白い方法がある．友人といっしょに次の実験をやってみなさい．英語の文書を1つ選び，あるところまで読み上げる．君が読むのを止めたとき，友人はその次の文字または記号を推測しなければならない．そこで君は次の文字または記号を見て，友人に正しいかどうかを告げる．友人が次の文字を正しく推測するまでの回数は，自由度の手軽な評価，すなわち次の文字に対する可能性の手軽な見積りになる．人間の推測は必ずしも正確ではなく，計算機による言語解析ほど良くはないが，これは1つの目安を与えてくれる．しかしどのような場合でも，人間は30分の1よりずっと良い確率で次の文字を推測できる！　事実，英語ではある文字につづく可能性のある文字の平均数は26ではなくおよそ5である．君たちは実験を何度も行ない，推測回数を平均すればこの数を算出できる．そしてこの実験によって英語がどれほど圧縮できるかの目安が得られる．また，これは1つのメッセージの中にどれほど多くの**情報**が存在するのかということをわれわれに考えさせる．この問題は後で取り上げる．

圧縮の問題を考えるもう1つの方法は，「それぞれが32の可能性を持っている N 個の英語の記号があれば，メッセージをいくつ送ることができるか？」と質問することである．「伝達できる情報の最大量はどれだけか？」でも構わない．すでに議論したように，32^N のほとんどは意味をなさないので，32^N の全部を送ることはできない．送ることが可能なメッセージ*7の数を n とする．このようなメッセージにそれぞれラベルを付けることができ，もし君たちが望めば数を使っ

*7　厳密に言えば，送ることができる，同程度に出現する可能性があるメッセージの数という意味である．実際には，いくつかのメッセージは他のものより多く現れる可能性があるだろう．しかし，これから厳密な議論をしないので，さしあたりこの点については悩まない．[RPF]

てラベルを符号化できる．この数として2進数を使おう．さて，どの数がどのメッセージに対応しているのかを記した長い一覧表を持った2人の男がいて，完全なメッセージを送って通信するかわりに，符号の数字を送って通信している．この状況は，「はい，私は散髪をする必要がある」や「いいえ，私は宿題をしなかった」などが書いてある両親の葉書を持っている君たちとまさに似ている．あるメッセージの全体を1つの数に圧縮するこの方法よりも効率的な送信法はないだろう．すべてのメッセージを送るためには何ビット必要かを計算できる．もしIビットが必要ならば，次のようになる．

$$2^I = n \qquad \text{または} \qquad I = \log_2 n \tag{4.21}$$

この数，すなわち英語(あるいは使われている符号体系が何でも)のNビットで伝えられるのと同じ量の情報を伝えるのに必要な最小のビット数を，送られる**情報量**あるいは単に**情報**と呼んでいる．この「情報」という用語の意味は，普通の用法と異なっていることを強調するのは重要である．まったく異なっているのではないが，一般的な意味での「情報」はメッセージの有用性やそのメッセージへの関心については何も表していない．ここで使う「情報」は厳密に学術的な用語である．科学にはこれに似た多くの用語がある．たとえば，物理学における「仕事」や数学における「関数」という用語の意味は口語での意味とほとんど関係がない．あとで情報という概念をもっと厳密に定義する．しばらくの間は，メッセージを2進数を使って符号化していることとメッセージを伝えるのに必要なビット数のぎりぎりの最小値に注目しているという，基本的な考え方だけを心に留めておいてほしい．

これまで発展させてきた考えを使って，「記号あたりの平均情報」を粗っぽく定義できる．N個ではなく$2N$個の記号があるとすると，これに対応する可能性のあるメッセージはいくつだろうか？　下の図に示すように，$2N$個の記号列をN個の記号の2つの列に分ける．

$$\overbrace{\cdots\cdots\cdots\cdots\cdots}^{2N} \!/\! \cdots\cdots\cdots\cdots\cdots$$

$$\underbrace{\cdots\cdots\cdots}_{N} \;/\; \underbrace{\cdots\cdots\cdots}_{N}$$

粗っぽい推定値として，送ることが可能なメッセージの数は，各記号列のメッセージの数の積，すなわちn^2になると期待される．もちろん一般には，正確な答を見つけるのはものすごく難しい．たとえば，「境界効果」とでも呼べる効果がある．すなわち2つのN個の記号の列は実際には分離されていないので，一方の列から他方の列へ境界を横切って単語が作られる可能性がある．また，列のある部分がある距離をおいて他の部分に影響を及ぼすという「長距離相関」がある．このようなことは英語では起こり，メッセージのある場所にある単語が存在すると，一般にはその近くにどのような他の単語が現れうるかに独特の影響を及ぼす．今までのところはまだ厳密ではないのでそのよ

うな問題は心配しない．事実，N をもっと大きくすれば，この粗い推定はもっと正確になる．もし $2N$ 個の記号があれば，およそ n^2 個のメッセージが得られる．$3N$ 個あればおよそ n^3 個，そして一般に xN 個の記号があればメッセージの数はおよそ n^x 個になる．N 個の記号による情報量を $I(N)$ と書くと，

$$I(xN) \approx \log_2(n^x) = x \log_2 n \tag{4.22}$$

となり，比

$$r = \frac{I(xN)}{xN} \approx \frac{\log_2 n}{N} = \frac{I(N)}{N} \tag{4.23}$$

は x に独立になることがわかる．したがって十分に大きな N については，われわれの近似が良くなっていく限り，この比は一定値に近づく．われわれはこの比を系の**記号あたりの情報**と呼んでいるが，その意味は式(4.23)の右辺から明らかであろう．

情報という概念にもどり，その意味をもっと良く理解するように努めよう．ある意味では，メッセージに含まれている情報量は，これを受け取ったときにわれわれが感じる驚きの量を反映している．たとえば「ご注文頂きありがとうございました．貴方が注文された本が在庫していることをお知らせします」，または反対の「申し訳けございませんが…は在庫しておりません」というような，印刷された通知を書店から受け取ることを考えてほしい．このような長いメッセージは，書店に直接電話したときに店員から聞き出す単なる「はい」とか「いいえ」に比べるとはるかに多くの記号を含むが，情報が多いわけではない．印刷された通知のほとんどの記号は，まったく余計なものである．彼らが言っていることを理解するには「ありがとう」とか「申し訳け」という単語を見つけさえすればよい．この点で情報は，メッセージの中に含まれているものであると同程度に，君たち自身の知的財産である．

この点を明らかにするために，誰かが君に 2 つの重複したメッセージ，つまり 1 つのメッセージとそのコピーを送ることを考えてほしい．彼から受信するたびに，同じものを 2 度受信する(これはエラー検出が目的ではなく，単なる悪い習慣である！)．ところで，2 つのメッセージに含まれている情報はそれぞれの情報の和にならなければならないというかもしれない($I(n_1n_2)=\log_2(n_1n_2)=\log_2 n_1+\log_2 n_2$ を思い出すこと)．しかしこれは間違いである．ただ 1 つのメッセージ，つまり最初のメッセージだけがあり，情報はこの部分だけから生じる．これは「情報」が単にメッセージの物理的な属性ではなく，メッセージの属性であり，かつそのメッセージについての君たちの知識量であることを例示している．

われわれの情報の定義と日常用語としての情報の違いを説明する最

良の方法は，ランダムなメッセージ，つまりNビットのランダムな2進数列を考えることである．もしすべての可能な2進数列が許容されるメッセージで，すべてのメッセージが同程度に出現すれば(もし各ビットに0または1が同程度に出現すればこうなるだろう)，そのようなメッセージに含まれる情報は，

$$I = \log_2(2^N) = N \qquad (4.24)$$

になるだろう．この値は，実はこのような記号を選択した場合に得られる最大の情報である．他のどのような型のメッセージも$I=N$にならない．さて，これは確かに意味をなさない．どうしてランダムな列が情報，それも最大量を含むことができるのだろうか？　きっと間違った「情報」の定義を使っているのだろうと君たちは考えるかもしれない．しかし考えてみると，前に議論したように，Nビットの数字列はそれぞれあるメッセージにラベルとして付けられるので，ある1つの数字列を受け取ると，われわれが受け取る可能性のある2^N個のメッセージのうちのどの1つを受け取っているのかがわかる．この意味で，この数字列には多くの「情報」が含まれている．メッセージを受け取ることによって，君たちはそれが何かを知らなかった状況からそれが何かを知っているという状況に変わる．そして受け取る可能性のあるメッセージの数が多いほど，そのうちの1つのメッセージを受け取ったときの「驚き」は大きく，または啓蒙されることも多くなる．受信前の不確定性と受信後の確定性の差が非常に大きいと言ってもよく，これが重要なことなのである．

4-6　情報理論

われわれは，メッセージに含まれる情報を次のように定義した．

$$I = \log_2 n \qquad (4.25)$$

ここでnは，われわれが受け取る**同程度の出現可能性がある**メッセージの数である．各メッセージには同じ量の情報が含まれている．一般の場合には，いくつかのメッセージは他のメッセージより出現可能性が高い．このような場合には，出現可能性の高いメッセージほど，その中に含まれている情報は少ない．メッセージに含まれる情報はそれを受け取ったときに体験する「驚き」を表すという主張を考えると，これは意味が通っている．この節の題は**情報理論**であるが，情報理論はわれわれが前に行なった考察を一般化するとともに，より厳密にすることができる．

まず簡単な例を取り上げよう．メッセージをあるアルファベットの記号から作ると仮定する．アルファベットとしては，DNAの4塩基やその他，いくらでもある．われわれは英語の文字に限定したくない．記号の種類の数をMとし，それらに次のようなラベルを付けよう．

$$a_1, a_2, \cdots\cdots, a_M$$

この言語によるメッセージは，これらの記号の，たとえばN個の長

い列になっている．さて，先に進む前に，これらの記号をメッセージ全体に分散する方法についてある仮定をしなければならない．まず各記号に確率 p_i を割り当て，これをメッセージの中で任意の記号が a_i である確率とする．量 p_i は a_i の出現頻度を示している．またメッセージ中の各記号は他のすべての記号と独立である．すなわち，ある位置にどの記号が現れるかは他の位置(たとえば直前)にある記号に依存しない．これは実際には，ほとんどの言語に対してまったく非現実的な仮定である．これが成り立たない場合についてすぐ後で考察しよう．

あるメッセージによってどれくらいの情報が運ばれるのだろうか？この答を求める簡単な方法は次のようなものである．われわれのメッセージの長さを N とする．すると記号 a_1 は平均して Np_1 回，a_2 は Np_2 回，…，a_i は Np_i 回見つかると期待される．N が大きくなれば，それだけこれらの推測は良くなる．異なったメッセージはいくつあるのだろうか？　組合せ理論の標準的な公式が助けてくれる．もし N 個の物があり，1つの型の物は m 個，別の型は n 個，また別の型は p 個，… ならば，$m+n+p+\cdots=N$ となり，その結果 $m, n, p, \cdots$ の可能な配列の数は次のようになる．

$$\frac{N!}{m!\,n!\,p!\cdots} \tag{4.26}$$

そこで平均としては，N 個の記号で表される異なるメッセージの数は次のようになる．

$$\frac{N!}{(Np_1)!(Np_2)!\cdots} \tag{4.27}$$

われわれは前に情報を，記号列の中の可能なメッセージの数の底2の対数であると定義した．この定義はすべてのメッセージに同程度の出現可能性がある場合にもとづいていたが，この定義は同程度の出現可能性でない場合にもよい定義であることを君たちは理解できる．したがって，あるメッセージ中の情報の期待値 $\langle I\rangle$ は式(4.27)の $\log_2$ をとることによって得られる．N が非常に大きいと仮定して，前に出てきたスターリングの公式を使うと次のようになる．

$$\langle I\rangle = N\sum_{i=1}^{M}(-p_i\log_2 p_i) \tag{4.28}$$

したがって，**記号あたりの平均情報**が得られる．

$$\langle I\rangle/N = \sum_{i=1}^{M}(-p_i\log_2 p_i) \tag{4.29}$$

この導出法は直観的なものであるが，もっと厳密に行なうことが可能である．シャノンはメッセージ中の情報を，そのメッセージが現れる確率の底2の対数と定義した．メッセージが現れる可能性が少なければ少ないほどそのメッセージの運ぶ情報はますます多いという，驚きとしての情報の考えと，記号あたりの平均情報との結びつきに注意し

てほしい．ある1つの特定の記号 a_i に含まれる情報は，明らかに，

$$-\log_2 p_i \tag{4.30}$$

であり，もしメッセージが n_1 個の a_1, n_2 個の a_2 などを含めば，その情報は，

$$I = -\log_2[(p_1{}^{n_1})(p_2{}^{n_2})\cdots(p_M{}^{n_M})] \tag{4.31}$$

で，これは，

$$-(n_1 \log_2 p_1 + n_2 \log_2 p_2 + \cdots + n_M \log_2 p_M) \tag{4.32}$$

となる．

ついでに言えば，もし2つのメッセージを連結すれば，このメッセージが伝える全情報は各メッセージが個別に伝える情報の2倍になることがこの式からわかるが，これは満足すべきものである．この点については君たちが確認してほしい．さてメッセージに含まれる平均情報は標準的な確率論を使って計算される．つまり

$$\begin{aligned} 平均情報 &= \Sigma(記号\ a_i\ の情報)\cdot(a_i\ が出現すると期待される数) \\ &= -\Sigma(\log_2 p_i)\times(Np_i) \end{aligned} \tag{4.33}$$

この結果は前に得られた結果である．ついでにいうと，シャノンはこの平均情報を「エントロピー」と呼んだが，これを大きな間違いだったと思う人がいる．というのは，これによって多くの人が情報理論と熱力学との関連を強調し過ぎることになったからである*8.

少しばかり違うが，このような考え方を説明するうまい例がある．かりにわれわれが電報会社で働き，等しくない確率で，「誕生日おめでとう」,「クリスマスは素晴らしいですね」というような，いくつかのメッセージを送るとする．これらのメッセージの各々には，客が注文する確率 P_m がある(たとえば $m=1$ から M)．ここで2つの型の情報を定義できる．まず電報を受け取る人が計算する確率がある──私の誕生日なので「誕生日おめでとう」というメッセージを私が受け取ってもあまり驚かないので，このメッセージに多くの情報はない．またこのメッセージを送るよう要求された電信技師が見たときの情報もある．2番目の型の情報を調べるのは興味深い．このために，電信技師の仕事の処理をしばらく調べ，そして彼らが送った各型のメッセージの割合を計算しなければならない．これによって送られているメッセージ m の確率 P_m がわかり，前に考察した親と学生の葉書の場合に各メッセージにラベルをつけたように，各メッセージをあるアルファベットの記号として取り扱うことができる．したがって電信技師の観点からその情報を計算できる．

4-7　さらに符号化技術について

さて，符号化の話題にもどって，メッセージを符号化する素晴らし

---

*8　言い伝えによれば，シャノンはこの用語を，数学者フォン・ノイマンの助言によって採用した．フォン・ノイマンは「ともかく誰も実際にエントロピーが何なのかがわからないので…この用語によってシャノンは論争で優位に立つだろう」と公言したということである．[RPF]

くそして巧妙な方法がいくつかあることを示すために，2つの人気がある符号化法を説明しよう．これらの符号は今まで考えてきたものとは異なり，記号の確率が多様なメッセージに対して設計されている．

4-7-1 ハフマン符号

8個の記号から成る次の系を考えよう．各記号の右にその出現確率を書いた(確率の降順に並べた)．

E	0.50
THE	0.15
AN	0.12
O	0.10
IN	0.04
R	0.04
S	0.03
PS	0.02

確率の合計は，ちゃんと1になっている．この系のメッセージの1例は，

ANOTHER

で，その確率は $0.12 \times 0.10 \times 0.15 \times 0.04$ である．さて，記号Eは他の記号よりずっと多く現れることに注目しよう．記号Eは記号PSの25倍も高い頻度で現れるが，PSは書くのに2倍の労力がかかる．この記号の系は非常に効率的であるとは思えない．効率が改善される新しい符号を書けるだろうか？「それでは，8個の記号があるので，3ビットの2進数符号を使おう」と素朴に考えるかもしれない．しかしEは高い頻度で現れるので，できれば，3ビットではなく1ビットだけで，これを表した方が良くないだろうか？ 他の記号を記述するのにもっと多くのビットを使わねばならないかもしれないが，これらの頻度はかなり低いのでまだいくらかの利益があるだろう．事実，われわれの符号よりも，メッセージの長さに関して，はるかに効率的な非均一な符号を設計できる．これは符号の圧縮の例になるだろう．モールスはこのように考えて「・」を頻度の高いEに，「— —・—」を頻度の低いQに割り当てた．

その考え方は，記号の長さを出現確率のほぼ逆数になるようにするというものである．すなわち最も頻度の高いものを1つの記号で表し，結果として典型的なメッセージの全長が短くなるように，記号の長さを変えるということである．実際にはこの記号を2進数列で置き換える．これからあらましを説明する技法はハフマンによるものである．君たちはこの技法がなぜうまくいくのか自分で考えてほしい．これはとても人気がある方法だが，私は多くの場合に実装するのは高くつくと信じている．これは2段階の過程で，次の枝分れ図を考えると最もよく理解できる．この図の第1列に，記号が確率の降順に配列してあ

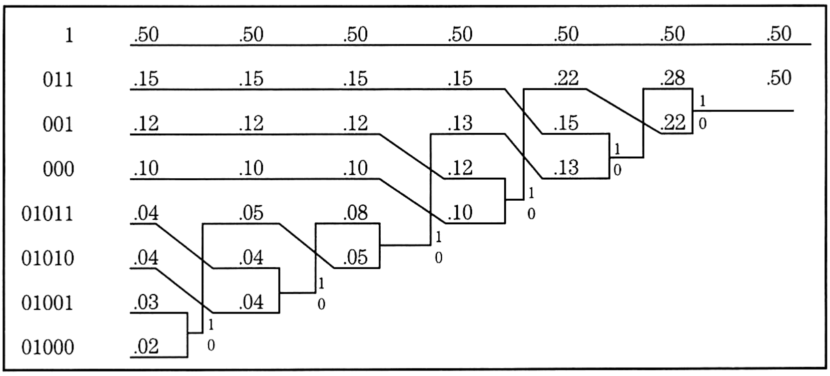


図 4-3　ハフマン符号化の枝分れ図

る(図 4-3).

確率の最も低い 2 つの記号を 1 つに合体させ，その確率を加えることから始める．さて，元の記号から下端の 2 つを除き，これに出現確率(この場合) 0.05 の新しい「結合」記号を加えた 7 つの記号のアルファベットがあると見なせる．図 4-3 のように縦の列を書き直し，結合記号をこの確率の階層構造の適切な場所に書く．そしてこの操作を繰り返す．すなわちこの一覧表をさらに縮めるために次の 2 つを合体させ，この調子で木の右端に到達するまでつづける．木の右端には，最初に最大確率であった記号と，その他の全部から形成された和の「結合」記号という 2 つの記号の「アルファベット」ができる．

符号化された記号の実際の割り当てを理解するために，ここで木の幹から枝の先まで経路を後もどりする．後もどりするときの規則は簡単である．元の記号にもどる経路の各分岐点で符号に数字を 1 つ加える．もしこの分岐点から上の経路をたどるなら 1 を，下の経路をたどるなら 0 を加える(これは純粋に規約の問題である)．木を右から左へ横切って移動するが，符号は左から右へ書き出す．図 4-4 に何が起こっているのかを示す．

「THE」に対する符号を調べよう．この符号を得るには，0 から始めなければならない．最初の分岐点では上の経路をたどる．ここまでで 01 となる．そして次の分岐点でも上の経路をたどらねばならない．結局，011 になり，これが THE の符号である．その他の符号は図に示されているようになる．ある点で結合した確率が元の確率の 1 つに等しくなる場合がたまに起こるという曖昧さを利用して，別のハフマン符号を作れるという事実は指摘する価値がある．結合したものと値の等しいものがある場合に表の中でどちらを上に置けばよいのだろうか？　考えてみたらどうだろう．

普通の 3 ビット符号の場合に比べて，この符号を使う場合に長さがどのくらい節約になるのかを計算するのはやさしい．3 ビットの場合には，1 つの記号の平均長は明らかに 3 である！　このハフマン符号

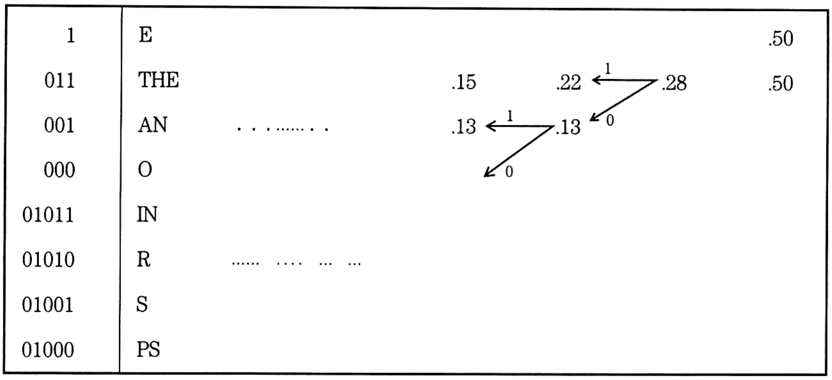


図 4-4　ハフマン符号化に対する「格子」

の場合には，記号の平均長は

$$(1\times 0.5)+3\times(0.15+0.12+0.10)+5\times(0.04+0.04+0.03+0.02)$$
$$=2.26$$

になるが，これは約4分の1の節約になっている！

長さが一様でない符号を構成する場合に，ハフマン法が利用しているすてきで巧妙な事実がある．これは**どの符号単語も，他のどの符号単語の先頭部分とも一致しない**という性質である．少し考えると，このような性質をもたない符号は役に立たないことがわかる．次のような記号の組，

1，01，10，101，010，011

があるとしよう．

メッセージ011010110を解読してみなさい．君たちには解読できない！　少なくとも一通りだけの解読はできない．これが01-1-01-01-10なのか，あるいは011-01-01-10なのか，あるいは01-101-01-10なのか，あるいはその他なのかはわからない．記号が互いに重なり合うことによる曖昧さがある．この困難を避けるために，ただ1通りだけの解読が可能な記号の選択が必要であり，ハフマン符号はまさにそのような符号である．われわれが見つけた符号の選択がただ1通りだけに解読されることを，君たちは確かめられる．

問題 4-4　ハフマン符号化法は，誤り訂正のためではなく圧縮のために開発されたという点でこれまでの符号化法とは異なっている．この方法によってうまく圧縮された符号が得られるが，この方法は誤りにはきわめて不安定である．次のようなメッセージ

00100001101010(＝ANOTHER)

は，1つの誤りによって変った記号に簡単になりうる．たとえば，2番目の位置が誤りならばTHEOTHERになる*9．この事実は興味深

＊9　とても多くの誤りがあると01011010010011になることもありうる．[RPF]（練習のためこれを解読してみてはどうですか．[訳者]）

い質問を投げかける.「長さが一様でない符号の場合，1 ビットの間違いによって生じる記号の誤りの最大数はいくつか？」 1 ビットの間違いの場合は 1 つの誤りだと考えることに慣れてきたが，長さが一様でない符号の場合には正しくないかもしれない. たとえば，1 つの誤りによって 1 つの記号が異なる長さの記号に変わり，そしてこれがその隣の記号に影響を及ぼし，そしてそのまた隣というように影響を及ぼしていき，そのために誤りがメッセージ全体に伝わることがありうるのではなかろうか？ そのようにはならない. その理由がわかりますか？

4-7-2　予測符号化

これまで，私は記号がメッセージの中に現れる確率が独立な状況だけを考えてきた. つまりメッセージの中の記号は互いに影響を及ぼし合わないと考えてきた. しかし，英語の例で強調したように，記号の相互依存性はきわめて一般的に現れる. 異なる出現確率をもち，相互に影響を及ぼし合う記号から構成されたアルファベットを完全に数学的に扱うのはきわめて複雑であり，私はここでは取り上げない. しかし，私はその影響によって起こる問題の香りをいくらかでも君たちにかいでもらいたいので，**予測符号化**を考えることにする. これは符号の訂正ではなく，符号を圧縮するもう 1 つの方法である.

もしメッセージの内容をあるところまで知っていれば，次の記号が何かを予測できるようなアルファベットがあると仮定しよう. この予測は確実な予測ではなく，確率的な予測で，どのようにして予測するかの詳細は問題ではない. われわれが使う方法は直前の記号の知識だけを要求してもよいし，直前 4 つの記号の知識，あるいはメッセージ全体の知識を要求してもよい. それは問題ではない. われわれは次の記号の予測を可能にするいくつかの規則を持っているとする.

さて，予測器を使って予測しよう. 予測器とは送信するメッセージの隣にくっつけるブラックボックスで，予測のための処方箋が入っている. 予測の仕方を説明しよう. 予測器はいままで送ったメッセージを完全に知っていて(メッセージの線を予測器に入力することによって図 4-5 に示している)，この情報にもとづいてどの記号が次に送られるかを予測する. それからこの予測(予測記号)は実際にやってくる記号と比べられる. もし予測が正しければ，送信線に 0 を送る. もし予測が間違っていれば，送信線に 1 を送る. これを実行する最も簡単な方法は，元の記号と予測記号をビットごとに加えて桁上げを無視するものである. 図式的には，送信端は図 4-5 のようである.

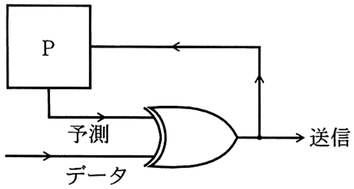


図 4-5　予測符号化器

予測が正しかったか否かを予測器に知らせるために，フィードバックループを組み入れていることを注意してほしい. 良い予測器は，と

ころどころに間違ったための1が散らばっている0の長い列を出力し，でたらめに推測するランダムな予測器は，メッセージの中の1と0の比率が半々であれば1と0を半々に出力する．予測器の出力を送信すると，同一の予測器を解読器として使用して，他端で元のメッセージをどのように再構成できるかを理解するのは難しくない．ただ逆向きに動作させればよい．もちろんこの方法はとても良いが，このかなり風変わりな処理の利点は何だろうか？　もし予測器がよくできていて，とても正確な予測をすれば，1の個数に比べてずっと多くの0を生成する．散在する1の間には0の長い列があるだろう．この方法の鍵は，予測器の出力をそのまま送るのではなく，**その中に含まれている0の個数を送る**ことである．もちろん，この数は2進数で送る．もし次の数字1の前に22個の0が並んでいたら，われわれは

0000000000000000000000

を送信するのではなく，22の2進数の

10110

を送る．これは送信の節約になる！　われわれが行なわねばならないのは，受信側の男にこの2進数を0の列に直させて，われわれが言おうとしていることを解釈するために彼の予測器を使わせることである．予測符号化によってメッセージを著しく圧縮できる．

問題 4-5　君たちが自分で楽める面白い問題は，もっと圧縮する方法を考えることである．0の列の平均の長さは，予測器がどの程度良くできているかによって決まる．予測器がどの程度良くできているかがわかると，ある0の列の長さがある長さになる確率を計算できる．そこで，もっと圧縮された符号を得るためにハフマンの技法を使うことができる！　出現可能性の等しい2進数の符号を送り，予測器が予測を誤る確率がqの場合について，その詳細を考えなさい．この種の圧縮を使うとシャノンの極限にかなり近づくことができる．

4-8　アナログ信号伝送

符号化の章の最後に，もう1つの符号化の問題を議論したい．これはもともとビットの形になっていない情報，すなわちアナログ信号をどのようにして送るのかという問題である．普通，車の油圧，駆動軸のトルク，金星表面の温度変化などの情報は**連続**である．すなわち，これらの量はどのような値でも取ることができる．もしビットを送信する能力しかなければ，この種の情報をどのようにして送るのだろうか？　これは基本原理の問題ではなく，実際的な問題である．これはやや周辺的な問題であるが，それでも私はこれについて少し述べたい．君たちはこのコース全体がやや周辺的だと言うかもしれない．だがちょっと待ちなさい！

まず手始めに，連続的な量，たとえばS，の変域が0と1の間だと

仮定しよう.

$$0 \leqq S \leqq 1 \qquad (4.34)$$

S の値を送る秘訣はこれを近似することである. 質問すべき最も重要な問題は, どの程度の精度でデータを送りたいかということである. S の精度を 1% 以内にしたいと仮定しよう. すると, われわれがしなければならないのは, 区間[0, 1]を 100 個のスライス(普通「ビン(箱)」と呼ぶ)に分割し, S の値がどのスライスの中にあるかについての情報, 0 と 99 の間の数を送信することである. これはやさしい. しかしわれわれは 2 進数を使いたいので, S の範囲を 128 のスライス($=2^7$)に分割し, S の値を 7 ビットの数で送る方が良い. 同様に, もし S を 1000 分の 1 の精度で送りたければ, [0, 1]を 1024 個のビンに分割して, 10 ビットの数を送るだろう.

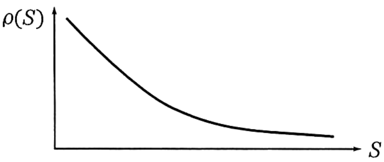


図 4-6　物理変数の確率分布の例

もし変数 S の取る値に制限がなければ何が起こるだろうか？　これは珍しいことではない. 普通, そのような変数は一様に分布していない値を取る. 言い換えれば, この変数は他の値よりもある値を取る確率の方が大きい(一様な確率分布を持つ物理量はきわめて稀である). 図 4-6 に示されているような確率分布を持つ変数があるとする. 確率密度 $\rho(S)$ は通常の定義である. すなわち, S の測定をしたときに測定値が S_1 と S_2 の間にある確率は

$$\int_{S_1}^{S_2} \rho(S)dS \qquad (4.35)$$

である. もし S_1 と S_2 が互いに無限に近くて, $S_2=S_1+\delta s$ ならば

$$\rho(S_1)\cdot\delta s \qquad (4.36)$$

である.

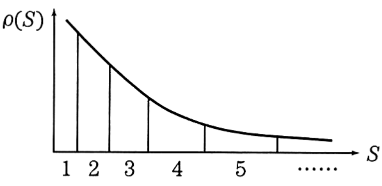


図 4-7　$\rho(S)$ の等面積のビンへの分割

この一般の場合に, S を送信する基本的な考え方は同じである. S が事実上意味を持つ範囲を多くのビンに分割するが, このとき各ビンの幅をビンが等確率になるようにするというのが重要な違いである(図 4-7). 明らかにビンの幅が違うが, 確率的に見れば, ビンが同じ面積になるように幅が選ばれている. ビンの面積は次の式によって定義されている.

$$\int_{S_i}^{S_{i+1}} \rho(S)dS \qquad (4.37)$$

ここで i は 0 から 127 までの値を取り, i 番目のビンは S_i から S_{i+1} の S の値に対応している.

別な方法として, 変数を変える変数変換もできる. S のすべての値 s に対して, 関数 $P(s)$ を次のように定義できる.

$$P(s) = \int_0^s \rho(S)dS \qquad (4.38)$$

$P(s)$ は S の累積確率関数で, $S \leqq s$ となる確率である. $P(s)$ が不等式 $0 \leqq P \leqq 1$ を満足するのは明らかである. この関数のよく知られた

統計的性質は(君たちが確かめられるとおり)分布が**平坦**だということである．すなわち，図4-6のSの関数のように，$P(s)$の確率分布を描けば，ただの水平な線になることがわかる．この結果，Pを等面積のビンに分けるとビンの幅は自動的に等しくなり，最初の場合にもどったことになる．

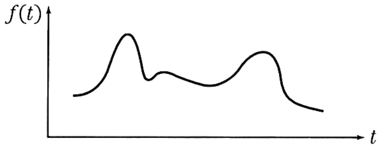


図4-8 典型的な時間の関数

異なっているが，関連している問題に時間の関数を送信する問題がある(図4-8)．このような問題を考えるときには，シャノンのもう一人の赤ん坊である有名な**標本化定理**を考えることになる．基本的な考え方は，ある一定な時間間隔，たとえばτで関数を標本化し，各時刻における関数の値をディジタルで送るというものである．受信者はこの離散的な数の組から元の信号を再構成しようとする．もちろん，一般の関数の場合には，受信者はこの数字の組を滑らかにして「ギャップ(途中の隙間)」を埋めなければならない．しかし，ある型の連続関数の場合には，この関数についての情報を**完全に**符号化するように標本化できる．すなわち受信者が元の関数を正確に再構成できるように標本化できる！　有限個の数によって連続関数をどのように記述できるのかを理解するために，フーリエ解析という数学的な問題を手短かに調べる必要がある．私はこの問題が面白いと思っているので取り扱うが，数学の予備知識のない人はここを飛ばしてもよい！

ある関数の「フーリエ変換」$g(\omega)$がすべての$|\omega|\geqq\nu$に対して0ならば(つまり$g(\omega)=0$)，この関数を間隔$\tau=\pi/\nu$で標本化すれば，これらの標本はこの関数を完全に記述する．この事実は何を意味するのだろうか？　フーリエ理論により，すべての周期関数$f(t)$は三角関数の項の和として表せる，という事実を思い出してほしい．一般的な時間の関数$f(t)$の場合には

$$f(t) = (1/2\pi)\int_{-\infty}^{\infty} g(\omega)e^{-2\pi i\omega t}\,d\omega \qquad (4.39)$$

となる．ここで$g(\omega)$は$f(t)$の**フーリエ変換**である．ここで実際に行なったのは，$f(t)$を適切な重みをつけて成分の周波数に分解したことである．さて通信理論で出会う典型的な関数(信号)には有限の**バンド幅**(帯域幅)がある．つまり，周波数に上限がある(たとえば，信号が送られるチャネル(通信路)は，ある値以上の周波数を伝えることができない)．このような場合には，式(4.39)の積分区間は有限になる．

$$f(t) = (1/2\pi)\int_{-W(\nu)}^{W(\nu)} g(\omega)e^{-2\pi i\omega t}\,d\omega \qquad (4.40)$$

ここでWはバンド幅で，νは$f(t)$のフーリエ展開の最高周波数である*10．

この式が，整数上の無限和になることが(その数学は少し難しいが)

*10　慣習ではバンド幅Wは$W=\nu/2\pi$で与えられる．［RPF］

示せる．

$$f(t) = \sum_{n=-\infty}^{\infty} f(n\pi/\nu)\cdot[\sin(\nu t - n\pi)]/(\nu t - n\pi) \qquad (4.41)$$

これが標本化定理である．この式を調べると，離散的な時刻

$$t = n\pi/\nu \qquad (n \text{ は整数}) \qquad (4.42)$$

での関数 $f(t)$ の値を知っている限り，他のすべての時刻での関数の値は，標本の信号による重みのついた項の重ね合わせとして求められることがわかる．これは式(4.41)でよく知られた関係

$$x \to 0 \text{ のとき } (\sin x)/x \to 1 \qquad (4.43)$$

を使った巧妙な結論である．すなわち，この和で $t=n\pi/\nu$ とおくと，n 番目の項以外のすべての項は消える．n 番目の項は 1 に $t=n\pi/\nu$ における f の値を掛けたものになる．言い換えると，ある関数を時間間隔が π/ν の時刻で標本化すると，われわれはこの標本から元の関数全体を再構成できる！　この結論は，関数 $f(t)$ が有限な区間 $(0, T)$ だけで定義されているという物理的に意味のある場合に最も興味深い．このとき式(4.41)は無限項の和ではなく，$f(t)$ が再構成できるためには有限個の標本点だけを取ればよい．標本点の数は $T\nu/\pi$ である．

標本化定理の数学的な証明は避けたが，この定理がどこから出てきたかの感触だけでも与えるのに時間を少し使うことは価値がある．われわれは，関数 $f(t)$ を等間隔 τ で標本化している．標本化された関数のグラフは，連続関数 $f(t)$ のグラフにスパイク状の「櫛」関数 $C(t)$ のグラフを掛けたものから得られる．この「櫛」関数は，標本点では 1，その他の点では 0 である(図 4-9)．

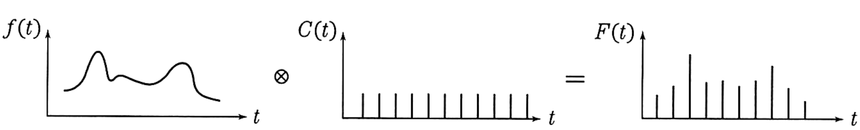


図 4-9　標本化された関数

さて，$f(t)$ に対応しているのはフーリエ変換の $\phi(\omega)$ である．$C(t)$ にもフーリエ変換があり，これは別の櫛関数 $\chi(\omega)$ である(図 4-10)．

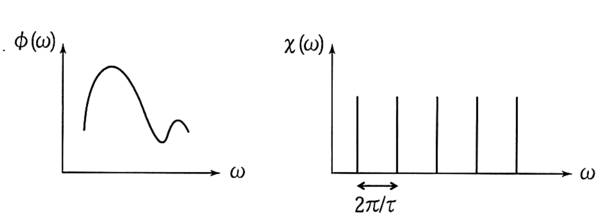


図 4-10　$f(t)$ と $C(t)$ のフーリエ変換

この変換 χ は，実は等間隔(間隔 $2\pi/\tau$)に並んだデルタ関数の集まりである．標本化された関数 $F(t)$ のフーリエ変換は「畳み込み」処理によって得られるが，これはグラフを使って粗っぽく説明すると，ϕ のグラフと χ のグラフの重ね合わせである．$F(t)$ のフーリエ変換は，

図 4-11 のように式(4. 41)の三角比によって高さの縮尺が決まり，水平軸に沿って等間隔に並んだ $f(t)$ のフーリエ変換のコピーから構成されていることがわかる．

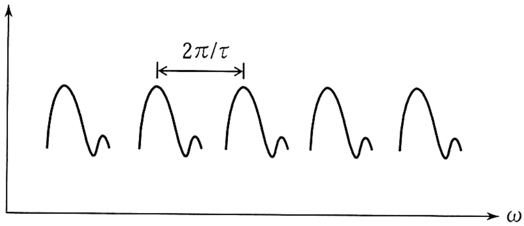


図 4-11 標本化された関数のフーリエ変換

このグラフをよく見てほしい．このグラフが示しているのは，$f(t)$ の全体についての情報は，原理的に $F(t)$ だけから引き出せるということである．図 4-11 のフーリエ変換の隆起 1 つの情報は，図 4-9 の全体の情報と同じ量がある！ 前者の変換は標本化された関数 $F(t)$ だけから生じるので，標本化定理の基本的な考えが現れていることがわかる．

興味深い微妙な現象が，標本化の処理でときどき起こる．標本化定理は，信号に周波数の上限 ν (すなわち $\nu/2\pi$ のバンド幅)があれば，少なくとも $T\nu/\pi$ 個の標本点が必要であることを示している．もしこれより多くの点を取れば，それらはすべて良い．しかし，もしこれより少なくしか取らなければ(これは，関数 $f(t)$ が区間 $(0, T)$ の外にたまたま「尻尾」を持てば生じうる)，標本化は不十分である．このような状況のもとでは**偽信号**(aliasing)と呼ばれているものが生じる．標本化の間隔が粗すぎて $f(t)$ の高周波成分を分解できず，その代りにこれらが低周波数の成分に入る──これが「偽信号」である．この現象の馴染み深い例は，映画の中に見つけられる．映画は 1 秒間に 24 回の割合で撮影した世界のスナップ写真という標本によって，われわれの目と脳に対して継ぎ目のない動きという幻想を作り出している．しかし，標本化したという形跡がときどき顔を出す．おそらく最もよく知られている例は，古い西部劇の荷馬車の車輪の動きである．私が子供の頃には，だれもが次のような現象に気づいていた．荷馬車が動き始めるとき，最初は車輪のスポークが正しい向きに回っているように見えている．スピードが増すと，車輪の回転が止まっているように見える．スピードがさらに増すと，車輪が逆方向に回転しているように見える！ この現象の説明は，標本化が不十分だということである．偽信号の別の例は，音のデータを不十分に標本化したときに起こり，われわれが普通聞くことのできない高い周波数が可聴周波数に偽信号化される．偽信号を避けるため，標本化の前に信号の高過ぎる周波数を濾波(フィルター)する必要がある．映画の場合には，これは写真をただぼやけさせるために，長い露出時間で撮ることを意味している．

現在では音を**ディジタル**，つまり 16 ビット，44.1 kHz で送り，完全に再生することが可能であり，ノイズにかなり強い．そしてそのような方法は，どのようなアナログ技術よりもはるかに優れている．そのような発展は将来を変える．映画も鮮明になるだろう．たとえば，光ファイバーによって**過剰**なほどの容量が得られる．石鹸の広告はとても鮮明に見えるだろう．技術的な世界は進歩するが，現実の人間の文化は泥の中で滑っているように思われる！

第 5 章
可逆計算と計算の熱力学

次に，自然界におけるきわめて興味深いが，まったく学問的な問題について考察したい．それは計算のエネルギー論である．われわれは「**計算を実行するのにどのくらいの量のエネルギーを使わねばならないか？**」という問題を考えたい．ところで，これはそれほど学問的とは思われない．結局のところ，現代のほとんどの機械の 1 つの特徴は，非常に速く動くときに，エネルギー消費がとても多いということであり，そして最も速い機械の制約の 1 つは，トランジスターのような部品が動作中に発生する熱を放出できる速度だからである．われわれがこの問題を「学問的」と述べた理由は，「計算を実行するのに**最小限**必要なエネルギーはいくらか？」というもう 1 つの基礎的な質問をしようとしているからである．

この問題のこのような物理的な面を紹介するために，前章で取り扱った分野，すなわち情報理論にもどろう．情報理論を厳密に物理的な観点から取り扱うことは可能であり，こうすることによって計算のエネルギーと結びつく．

5-1　情報の物理

まず君たちに，メッセージに含まれている情報量の物理的な定義を理解してもらいたい．物理学がこの領域に関係することは驚くに当たらない．初期にシャノンは実際の導線でメッセージを送ることに関心を持っていたこと，そして物理的世界とまったく関わることなしにどのような種類のメッセージも送れないことを思い出してほしい．送られるメッセージの非常に基礎物理的なモデルを考察することによって，これらの事柄を説明しよう．

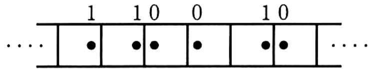


図 5-1　基本的な原子のメッセージ

図 5-1 に示されているように，入ってくるメッセージは，それぞれの箱の中に 1 個の原子が入っている，一連の箱だと心に描いてほしい．各々の箱の中で原子は左側と右側の 2 つの場所のうちのどちらかに存在できる．もし原子が左側に存在すれば 0 ビットとみなし，右側に存在すれば 1 とみなす．このような箱の流れが私のところにやってきて，通り過ぎていく．各原子が箱の中のどこに存在するかを見ることによって，対応するビットを知ることができる(図 5-1)．このモデルが，どのようにしてわれわれが情報を理解する手助けになるのかを知

るためには，上下左右に急速に動いている原子の物理学を考察せねばならない．そのためには気体の物理学を考察せねばならないので，気体の物理学の若干の事項から話を始めたい．

N 個の原子(または分子)を含み，体積 V_1 を占める気体があると仮定しよう．この気体はきわめて単純なものだとして，その中にある原子または分子は(ここでは原子と分子という2つの単語の区別をしない)本質的に自由であり，各原子の間に引力も斥力も働かないものとする(これは適度の低圧ではよい近似である)．さて，この気体をピストンで押して，体積が V_2 になるまで圧縮する．この過程を一貫して等温的に行なう．つまり，全系を一定な温度 T の「熱浴」に浸し，この装置の温度が一定であるようにする．このようなことがわれわれの話題に関係があるとすれば不思議ではないだろうか？　これからどのように関係しているのかを示そう．まず気体を圧縮するのに必要な仕事 W がどれだけになるかを知りたい(図5-2)．さて力学の標準的な結果によれば，力 F が小さな距離 δx だけ移動するときになされる仕事[*1] δW は

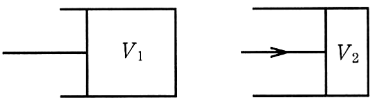


図5-2　気体の圧縮

$$\delta W = F\delta x \tag{5.1}$$

である．気体の圧力が p で，ピストンの断面積が A ならば，$F=pA$ を使い，気体の体積変化を $\delta V = A\delta x$ と表すと，式(5.1)を次のように書き換えることができる．

$$\delta W = p\delta V \tag{5.2}$$

さて，気体論の標準的な結果を利用しよう．圧力 p，体積 V，温度 T の理想気体には次の関係がある．

$$pV = NkT \tag{5.3}$$

ここで N は気体中の分子の数で，k はボルツマン定数(約 $1.381 \times 10^{-23}\ \mathrm{JK^{-1}}$)である．等温過程だと仮定したので T は一定で，W を求めるための積分は簡単に実行できる．

$$W = \int_{V_1}^{V_2} \frac{NkT}{V} dV = NkT \log \frac{V_2}{V_1} \tag{5.4}$$

(ここで，$\log x = \log_e x$)．V_2 は V_1 より小さいので，この量は負になるが，これは，気体が行なう仕事ではなく，気体になされた仕事の場合は負符号を持つという規約の結果にすぎない．さて，気体を圧縮すると気体はふつう加熱される．これは構成原子の速さが増し，運動エネルギーが増えた結果である．しかし，われわれの場合には，圧縮の前後で気体分子を調べると，何も違っていないことがわかる．同じ数の気体分子があり，分子は前以上でも前以下でもない勢いで動きまわっている．2つの状態間には分子のレベルでの相違はない．それではあの仕事はどこへ行ったのだろうか？　われわれは気体を圧縮するた

---

*1 「情報」のようなやっかいな別の言葉の1つ．この定義では，力は仕事をするためにある距離だけ移動しなければならず，そのためにスーツケースを持ち上げたままでいてもこの種の「仕事」にはならないことに注意すること．持ち上げることだけが仕事になる！[RPF]

めにいくらかの仕事をこの気体につぎ込んだので，エネルギー保存のためにこの仕事はどこかに行かねばならない．実際には，この仕事は気体の内部熱に変換**された**のだが，すぐに，この気体を等温に保っている熱浴に流出した．これが等温圧縮の実際の意味である．つまり気体とその周囲の熱浴とが常に熱平衡になっているように，気をつけながらゆっくりと圧縮するのである．

熱力学の観点からは，われわれが行なったことは，体積 V_1 を占める気体から体積 V_2 を占める気体への「状態変化」である．この過程で，気体の構成粒子のエネルギーの和である気体の全エネルギー U は変化しない．このような状態変化を議論するのに適した熱力学量は，**自由エネルギー** F と**エントロピー** S であり，これらには次の関係がある．

$$F = U - TS \tag{5.5}$$

自由エネルギーは，2つの状態間に力学的な違いがなくても，それらの違いを議論できるようにするために考案された概念である．自由エネルギーの意味をもっと良く理解するために，等温微小変化の場合の式(5.5)の

$$\delta F = \delta U - T\delta S \tag{5.6}$$

を調べよう．ここで考えている変化では，気体の全エネルギーは一定なので，$\delta U=0$ であり，$\delta F=-T\delta S$ となる．δF は熱浴へ吸い込まれた熱エネルギー「損失」，$NkT\log(V_1/V_2)$であり，これを使うと，式(5.6)からエントロピーの変化が，

$$\Delta S = Nk\log\frac{V_2}{V_1} \tag{5.7}$$

と表される．ここでは有限の変化を扱っているので，無限小の δ を有限の Δ で置き換えたことに注意すること．

エントロピーはかなり奇妙で直観に反している量なので，これに焦点を当てるのがよいのか，それとも自由エネルギーに焦点を当てるのがよいのかについて，私はいつも迷う！　熱力学を少し知っている人は，この標準的な公式 $\delta S=-\delta F/T$ は，絶対温度 T の系に熱量 δQ が入ったり出たりする熱力学的に可逆な状態変化の各段階で生じるエントロピーの無限小変化に対する標準的な公式 $\delta S=\delta Q/T$ の別の形であることがわかるだろう．不可逆過程に対しては，等号は不等号で置き換えられ，孤立系のエントロピーが一定か増加するだけかであることを保証しているが，——これが熱力学の第2法則である．すぐ後でエントロピーについて少し述べる．

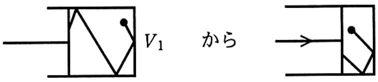
さて，少し飛躍する．飛躍できることは明らかではないが，飛躍する．この気体が1分子だけを含む場合を考える．つまり，式で $N=1$ と置く．さて1分子だけが存在するときには，自由エネルギーやエン

トロピーはいうまでもなく，温度，圧力および体積のような概念についての感じをつかむのも難しい！　しかし，これらの量の時間平均を考えて，この1粒子が前後に跳ね返るときの不規則性を取り除けば，これらの概念は意味をなす．事実，この隠された不規則性の除去がある限り，この式は$N=1$でも成り立つ．この状況を考えるのは面白い！

この分子によって占められる体積を半分にして，$V_2=V_1/2$にしてみよう．そうすると，この粒子の自由エネルギーとエントロピーはそれぞれ次の量だけ変化することがわかる．

$$+kT\log 2 \quad \text{および} \quad -k\log 2 \tag{5.8}$$

これは何を意味するのだろうか？　図で表すと，状況は，



に変わる．ところで，圧縮の前後での分子の物理的状態は同じように見える．たとえば，分子の運動エネルギーは変わっていない．しかし，ある理由でFとSは変化している．起こったことはとても微妙なことだが，**この分子が存在可能な場所についての私の知識が変化した**ということである．初期状態では，分子は体積V_1のどこかに隠れているが，圧縮の後では体積V_2の中のどこかにいなければならない．言い換えると，分子の存在可能な場所が減少したのである．

この「知識」という概念はきわめて重要で，エントロピーの概念にとって中心的なものなので，しばらくこれについて詳しく述べたい．これは，熱力学とはそもそも統計的なものだという性質から生じている．気体を構成する膨大な数の粒子を数学的に扱うときに，気体のすべての分子の経路と運動量を追っていくことは実際には不可能なので，そのため確率論に頼らざるをえない．気体の温度や気圧のような概念は，本質的に，統計的な平均として定義される．すなわち，各分子のある属性を考え，この属性についてこれらの分子が特定な分布をしていると仮定し，ある重みをつけて平均する．つまり，多くの分子はある温度に対応するある速さで動き，別の多くの分子は別の温度に対応する別の速さで動き，われわれはこれらのすべてについての平均をとる．

気体のエントロピーも統計的に定義される．この定義は確かに核心になる定義であるが，温度やエネルギーのような量の定義とは異なる仕方で定義される．これらの量とは異なり，エントロピーは微視的な性質の和から生じる巨視的な性質ではない．むしろ，**気体がこの配位(configuration)にあることが見つかる確率**に直接関係している．私は「配位」という用語によってN個の構成分子の各々の位置と運動量の特定の配置，すなわち一群の配置(あるいは気取った言い方をす

ると「位相空間」のある点，または領域)を意味している．このような確率の存在はそれほど驚くようなものではない．たとえば，ある気体を調べると，すべての分子が同じ方向に動いていたり，対になってダンスするように飛んでいたりするのを見つけることは，分子がいたるところでランダムに素早く動いているのを見つけることに比べてはるかにありそうもない．エントロピーはこの考えを定量化する．おおざっぱに言えば，もし気体の特定の配位の確率が W なら，次のようになる．

$$S \approx k \log W \tag{5.9}$$

W が大きければ，エントロピーは大きくなる．そして，他のすべての確率のように W の加算ができる．したがって，配位がある範囲にある可能性が簡単に計算できる．全部の分子が 1 つの向きへ動いている気体の W は，もっとランダムな構成，すなわち**もっと無秩序な**構成の気体の W よりずっと小さくなり，したがって小さなエントロピーを持つ．このようなことがある系についてのわれわれの知識と何の関係があるのだろうか？　簡単にいうと，気体の配位についての知識が少ないほど，気体はより多くの状態に存在可能で，全 W は大きくなり，エントロピーが大きくなる．これが気体を小さな体積に圧縮するときに起こることについてのよい直観的理解を与える．等温圧縮では容器内の分子の運動量は同じままであるが($\delta U=0$)，各分子は空間的位置の可能性の少ない状態になっていく．したがって，この気体は小さな W を持つ配位になり，そのエントロピーは減少する．ついでに記すと，熱力学の第 2 法則によれば，任意の孤立系で

$$\delta S \approx k\delta W/W \geqq 0 \tag{5.10}$$

である．すなわち，エントロピーは決して減少しない．圧縮された気体のエントロピーが減少したという事実は，系は孤立していないことを示す──われわれは熱浴へ熱を流出させてきた．熱浴への熱の流入は熱浴のエントロピーを増加させ，第 2 法則を成り立たせる．一般的に言えば，**ある状態についてわれわれが持つ情報が少ないほど，エントロピーは大きい**．

エントロピーの定義は本質的に統計的なので，これまで考えてきたような 1 分子の気体に対してエントロピーを定義することに何の問題もない．ただし，微妙なところが若干ある(が，われわれはこの問題を敬遠する)．もし体積を 1/2 に圧縮すれば，空間位置の数，したがって分子が占めることのできる配位の数は半減することがわかる．圧縮前には，分子は箱のどちらかの半分の中にも存在可能であったが，圧縮後には片方の半分の中にのみ存在可能である．君たちは，この事実がどのようにしてエントロピーを量

$$\delta S = k \log 2 \tag{5.11}$$

だけ減少させるのかを，確率的な描像で理解できるはずである．この結果は，われわれが仕事と自由エネルギーの考察で求めた結果と同一

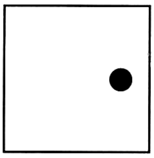
である．

ここで，情報の話題にもどると，この不思議な物理学がどこにうまく当てはまるかがわかる．この節の始めに示した，箱の中の原子の位置がメッセージの2進数のビットを表すという原子のテープを思い出してほしい．さて，このメッセージが典型的なものとしよう．つまりこれらの数字のいくつかについてわれわれはあらかじめ知らないが，他のビットについては──前もってそれらを知っているか，あるいは前に調べたビットとの関連から知ることができるために──知っているとする．メッセージに含まれている情報を，**全テープを0にリセットするのに必要な自由エネルギー量に比例する**ものとして**定義**しよう．「0にリセットする」とは，構成原子が「0」位置にあることを保証するために，テープの各セルを圧縮するということである．

この定義には明らかに問題だと思われる点があること，すなわち，この定義には0と1を非対称に扱っているという不自然さがあることにすぐ気づく．もし原子がすでに区画の0の部分にあれば，リセットの処理の操作は不要なので，自由エネルギーを必要としない．一方，原子が区画の1の位置にあれば，0の位置へ移動させるために仕事をしなければならない！　この事実が意味をなすとは思えない．このテープを「1にリセットする」ことにすると情報の別の定義ができると思う人がいるだろうが，こうするとメッセージが1と0を同数ずつ含んでいる場合にのみ同じ答が得られるだけなように思われる！　しかし，ここには微妙なところがある．区画のどちら側に原子が存在するのかをわれわれが**知らない**場合にのみ，われわれは自由エネルギーを費やすのである．原子の位相空間が半分になり，エントロピーが減少するのはこの状況においてのみである．もしわれわれが原子の位置を知っていれば，この原子の最初の位置に関係なく，リセットをするのにわれわれはエネルギーを費やさ**ない**．言い換えれば，われわれが望んでいるように，メッセージに含まれている情報はびっくりさせるビットに含まれている．なぜそうなるのかは詳しく話す価値がある，というのはこの説明には可逆計算の世界でしばしば見られる議論のスタイルが関わっているからである．1を0へリセットするのに必要なエネルギーは，0のままにしておくのに必要なエネルギー，言い換えれば何もしない場合に必要なエネルギーと同じになると断言することは少し直観に反しているように思われる．

この点を明確にするために，まずわれわれが考えている装置の理想化されている性質を強調しなければならない．私は「箱」の中にある原子について大まかに話してきたが，これらの箱は，ボール紙でできていて糸でつないであり，質量と運動エネルギーと位置エネルギーを持つというような，実際の箱ではない．さらに，私が「エネルギー」について話すときには，私はテープのエネルギーという意味で言って

いるのではない！　われわれは原子の位置によって指定されているメッセージの内容にのみ関心がある．さて，中身が1だとわかっているメッセージビットがあるとしよう．つまり原子は右側にあるので，次のような描像になる．



これを0にリセットするためにエネルギーは必要でないことをいくつかの方法で示すことができる．1つのかなり抽象的な方法は，まず，原子をある場所に留めるために小さな隔壁を挿入することである．そうしておいて，次に私がしなければならないのは**箱をひっくり返す**ことだけである．そうすると右側に原子はなくなる(図5-3)．エネルギーを費やすことなく隔壁を挿入して箱をひっくり返せるのは奇妙に思われるので「抽象的だ」というのである．もちろん，実際の世界ではこのようなことはできないが，ここでは抽象化を行なっているのであって，私が言ったように，われわれは「箱」の運動エネルギーや重さに興味はない．仮定が満たされればそうすることができるのである．ただ，がっかりさせるのは，これを行なうためには無限に長い時間をかけねばならないということである．(われわれは5-2節でこの種の議論にもどる．）これより少しは抽象的でない，もう1つの方法は，箱の両側に1つずつ，合計2つのピストンをつけて，原子を1つのピストンで押しながら，他方のピストンを引き出すことである(図5-4)．さて，左側のピストンへの衝撃は右側のピストンへの衝撃と同じであり，一方で外からなされた仕事は，すべて他方で外へする仕事になり，したがって相殺される．2つのピストンを箱の上方にある棒に結合させて連結すれば，1つのピストンにそっとさわるだけで両方のピストンが最終位置まで動いていくことがわかるはずである．したがって，もしこれを十分にゆっくり行なえば，「無限に遅い極限」では，リセットの際に仕事はなされない．テープをクリアすること，つまりリセットすることは，われわれがこの原子がどの区画にいるのかを知らない時に起こることである．このときには，われわれは圧縮しなければならないが，以前に議論したように，これは原子の位置についてのわれわれの知識を増やすので，この過程では自由エネルギーが増える．

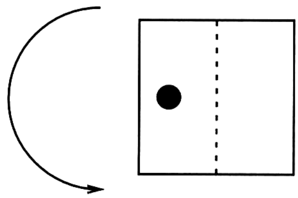


図5-3　単純なリセットの処理

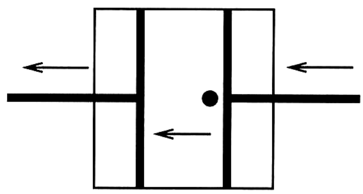


図5-4　より「現実的な」リセット

これらの考え方を考察するもう1つの方法は，ベネットによるものである．彼はメッセージテープを**燃料**として使うことを提案し，テープの中の情報と燃料値，つまりテープから得られるエネルギーの量とを関係づけた．彼の考え方はとても巧妙で，次のようなものである．ある種の熱浴と接触している機械があり，この機械は一端でテープを取り込み，他端でテープを出すものとする．まず初めに，この機械が

取り入れるテープは空白だと仮定する．すなわち，テープのすべての原子は 0 の状態にあると仮定する．このようなテープをどのように使えばこの機械の動力として使える仕事を供給させられるかを示そう．

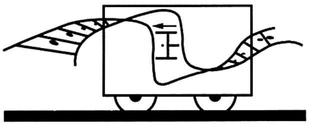


図 5-5　情報駆動エンジン

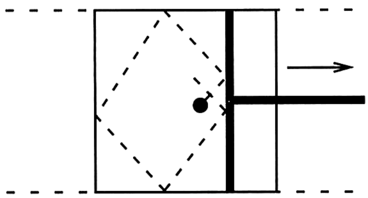


図 5-6　エンジン内での仕事の生成機構

われわれはこの系にピストンを組み入れる．各々の箱が入ってきたときに，われわれはピストンを箱の中に持ち込み，箱の半分の位置まで入れる(図 5-5)．次に熱浴でこの箱を温める．そうすると箱の中の原子はピストンに向って突進し，図 5-6 のように等温的にピストンを外側へ押す．これはこの節の初めに考えた気体の圧縮の逆過程である．正味の結果は，取り出せる仕事がピストンになされたということである．言い換えれば，このテープが仕事をするようにできるということである．T を熱浴の温度とすると，n ビットのテープの場合に，この仕事は自由エネルギー $nkT\log 2$ に等しいことが君たちにわかるはずである．われわれの処理の重要な結果は，機械が出力するテープは**ランダム化**されているということである．つまり，ピストンが押し出された後，押した原子はその箱の中のどこにでも存在可能で，測定しなければ原子がどこにいるかを知る方法はない．

次に，このピストンの場所が変えられると仮定して，この議論を一般化する．そうすると，1 が 1 つあるテープから仕事を引き出すことが可能になる．1 がやってくると，われわれはピストンを箱の反対側に持っていき，箱の半分の位置まで入れ，前と同じようにつづける．この場合にも，われわれは役立つ仕事 $kT\log 2$ を手に入れて，そしてこの機械から出てくるテープはやはりランダム化されている．ここで決定的なことは，0 と 1 のどちらがいまこの機械に入るところなのかを，われわれが**知っている**ということである．この場合だけ，ピストンが確実に仕事をするようにピストンを準備できる．もしピストンを 0 の位置にしておいて，1 が入ってきたら，原子を 0 の場所へ移動させるためにわれわれは実際に仕事をしなければならず，原子が膨張して箱全体へもどるときにわれわれはその仕事を取りもどすことになるのは明らかである．つまり，役立つ仕事はなされない．明らかに，**ランダム化されたテープの燃料値はゼロである**．もしどの数字が次に入ってくるのかを知らなければ，われわれはこのピストンをどのようにセットすればよいのかわからない．そこでこのピストンを一方の位置のままにしておき，これを真ん中まで入れて待つ，ブーン，ブーン，ブーン．時には運が良くて，ある原子がピストンを押し出して仕事をしてくれることがあるが，完全にランダムなメッセージの場合には，同じ程度の可能性でわれわれは原子に仕事をしなければならない．その結果，われわれの機械に動力を供給する正味の仕事は，ゼロである．

明らかに，ベネットのテープの機械はわれわれのリセット過程と逆のことを行なっている．彼は仕事を引き出すためにメッセージテープ

を使い，ランダムなテープにして終わる．これに対して，われわれはランダムなテープを取り入れてこれに仕事をして，標準的な，全部が0のテープにして終わる．この逆関係はベネットの枠組での情報の定義に反映されている．Nビットのテープがあるとしよう．われわれはこのテープに含まれている情報Iを式

$$\text{テープの燃料値} = (N-I)kT\log 2 \tag{5.12}$$

で**定義する**．この式から，われわれに満載の燃料，つまり1ビットあたり$kT\log 2$の燃料を与えるテープの運ぶ情報は0であることがわかる．そのようなテープは完全に予測できる内容を持っていなければならないため，この結果は，われわれが期待していたものである．これらの2つの方法の間には魅力的な物理的対称性がある．あるメッセージテープをこの機械に通すと，このテープからあるエネルギーEを引き出せるだろう．ところで，このエネルギーEは新しくランダム化されたテープを元の形にリセットするのに必要なエネルギーに正確に等しいだろう．もちろん，これらの事柄について考えるとき，どちらの描像を採用するかは君たち次第である．私は，情報を得るためにいつもNから引かなければならないのは好きでないという理由もあって，リセットの描像の方を選ぶ．

君たちに，ベネット博士の機械についていくつかの問題を熟考してもらいたい．

問題 5-1　Nビットのランダムなテープと正確なコピーの2本のテープがあると仮定する．2つのテープを合わせたものの燃料値が$NkT\log 2$であることを示せる．この2つのテープから，この量のエネルギーを引き出せる機械を設計できるかどうか調べなさい．（ヒント：1つのテープを他のテープに対して「相対的に」拡張させねばならない．）

問題 5-2　3ビットが連続してくり返されている，たとえば110110110110⋯のようなテープがある．$3N$ビットのテープの場合，燃料値はいくらか？　どのようにしてその値を求めたか？

5-1-1　マクスウェルの魔と測定の熱力学

情報についての物理学をもっと勉強したい人にとって最良の勉強法は，偉大なスコットランドの物理学者ジェームズ・クラーク・マクスウェルが発見した19世紀のパラドックスについての多くの参考文献を調べることである．**マクスウェルの魔**(デモン)と呼ばれているものは，物理学者の間に1世紀におよぶ大論争を引き起こした．この論争が解決したのはつい最近である．ベネットとランダウアーのような研究者が可逆計算についての結論や，計算のエネルギーを導くのに部分的に寄与し，情報とエントロピーとの関連を明確にしたのはマクスウ

ェルの魔についての熟考によってであった．このような研究は，これらの問題における**測定**の役割に光を当てた．これは重要なことである．私はここではこの問題に詳しくは立ち入らないが，君たちに興味を持たせるのに十分なさわりの部分を紹介しよう．マクスウェルの魔，およびこれを理解する試みについての十分な議論は，H. S. レフと A. F. レックスの Maxwell's Demon : Entropy, Information, Computing (Adam Hilger, 1990)に記されている．

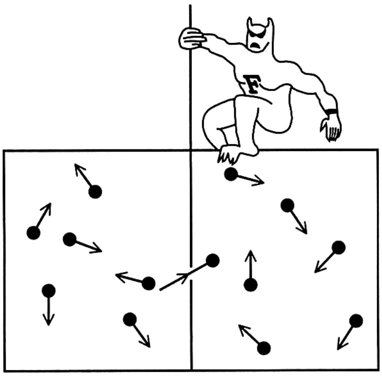


図 5-7　仕事中のマクスウェルの魔

マクスウェルが想像したように，仕切られた箱の上に小さな魔が座っていて，箱の各半分は位置と速度がランダムに分布している分子の気体で満たされていると想像しよう(図 5-7)．この魔の職務はとても簡単である．箱の仕切りには弁がついていて，魔は思い通りに弁を開閉できる．彼は箱の半分(たとえば，左側の半分)を眺め，速く動いている分子がこの弁に近づいてくるのを見つけるまで待つ．速い分子を見つけると，魔は一瞬間だけこの弁を開けてこの分子を右側に通し，そして再び弁を閉じる．同じように，魔は遅く動いている分子が右側から弁に近づいてくるのを見つけると，それを左側へ通す．ある時間が経つと，このような活動によって，われわれの小さな友人は速い分子と遅い分子を，2つの区画に分離してしまうだろう．言い換えれば，彼は熱い分子と冷たい分子を分離し，したがって，箱の仕切りの両側に温度差を生じさせるだろう．これはこの系のエントロピーが減少したことを意味し，明らかに第2法則に反する！

このパラドックスのように見えるものは，私が言ったように，物理学者の間にものすごい論争を引き起こした．熱力学の第2法則は物理学の十分に確立された理論であり，もしマクスウェルの魔がこの法則に反することができるように見えるなら，おそらくこの魔には何か疑わしいことがある．マクスウェルが1867年にこの考えを思いついて以来，多くの人たちが彼の議論の欠陥を見抜こうとした．あるタイプの分子を見つけて弁を通すという過程で，どこかで何らかの方法でエントロピーが生成されねばならない．

最近まで，魔が分子の位置を**測定**した結果としてこのエントロピーが生じたという説が広く受け入れられていた．これは非合理だとは思われなかった．たとえば，この魔が速く動く分子を検出できる1つの方法はこれらの分子を魔法のランプで照らすことである．しかしこのような過程には少なくとも1個の光子の放出が伴っていて，この放出にエネルギーが費やされる．もっと一般的には，特定の分子を見る前に，魔はこの分子が左と右のどちらに向かって動いていたのか知りえなかった．しかし観測によってこれを知り，彼の不確かさ，したがってエントロピーは半減し，一方，それに伴って，環境の中に対応する量のエントロピーが生成した．

驚くべきことに，マクスウェルの魔がすべての情報の記録と消去を

ある規則に従って行なえば，この魔はエネルギー消費がゼロでも実際に測定を行なえることをベネットは示した．この魔は測定の前にある種の標準状態にいなければならない．この標準状態をSと呼ぼう．これは不確定の状態である．魔がある分子の運動の向きを測定すると，魔は他の2つの状態のうちの1つになる．「左へ動いている」場合はL，「右へ動いている」場合はRである．魔はSをLかRのどちらか適切な方で上書きする．ベネットは，エネルギーを費やさずにこの処理を実行できることを示した．次の測定に備えるために魔をリセットして状態Sにするために，LやRを**消去する**という次の段階でエネルギー消費が入ってくる．計算過程におけるエントロピーの生成源が，測定ではなく情報の消去であるというこの認識は，可逆計算の研究における飛躍的な前進であった．

5-1-2　エネルギーとシャノンの定理

物理学と情報という話題を終える前に，前の章で学んだ，ある通信路で送れる情報量の限界という問題にもどりたい．手に入れた物理学の道具を持ってシャノンの定理を再訪できることは，君たちにとって驚きではないだろう！　情報の物理学についての研究と，われわれが以前に行なった誤りについての勉強とを結びつけよう．1つの興味深い質問は，「メッセージに1つの誤りが出現すると，そのメッセージの情報量にどのような影響を及ぼすか？」である．あるメッセージの全Mビットを完全に知っている場合から始めよう(このメッセージの情報量をNとする)．われわれはこのメッセージを雑音の多い通信路を通して送ろうとしている．この影響によって，受信されたメッセージの各ビットは確率qの割合で誤りとなって届く．おなじみの「Mビットの情報を正しく伝えるために送らねばならない最小ビット数はいくつか？」という質問をしよう．われわれはメッセージを符号化せねばならず，この質問を以前に考察したときに使った記号をここでも使うと，符号化されたメッセージは長さM_Cになっている．これはわれわれが実際に送るビット数である．さて，テープの内容について何も知らないとすると，このテープを消去するために，次に示す量の自由エネルギーを消費する必要があることをわれわれは前に示した．

$$M_C kT \log 2 \tag{5.13}$$

しかし，このエネルギーのうちのいくらかは誤りを消すことに費やされる．以前の導出法を使うと，この部分は平均では次のようになる．

$$\begin{aligned} &M_C kT \log 2[-q \log_2 q-(1-q)\log_2(1-q)] \\ &\quad =[1-f(q)]M_C kT \log 2 \end{aligned} \tag{5.14}$$

われわれはこれだけのエネルギーが無駄に費やされると考える．この結果，メッセージを消去するために費やされる自由エネルギーは，

$$M_C kT \log 2-[1-f(q)]M_C kT \log 2=f(q)M_C kT \log 2 \tag{5.15}$$

になる．そこで，自由エネルギーと情報の関係を使うと，エネルギー保存によって，この通信路を通して送れる情報の最大量は，

$$M_C[1-q\log_2(1/q)-(1-q)\log_2(1/(1-q))] \qquad (5.16)$$

となる．この種の物理学的議論からどのようにしてシャノンの結果が導かれるのか，を君たちはわかるだろう．

5-2　可逆計算と計算の熱力学

計算のすべての段階でエネルギーが費やされるということはこれまで常に仮定されてきた*2．最初の推測であり，長い間広く信じられてきた推測は，機械によって行なわれる各論理操作には必要なエネルギーの最小量が存在するという推測であった．今までわれわれが考察してきたことから，君たちはこの議論を理解できるはずである．この考え方は，ある装置のすべての論理的状態はその装置のある物理的状態に対応していなければならず，AND ゲート中のトランジスターのように，この装置が出力として 0 と 1 のどちらかを選ばねばならないときには常に，この物体の利用できる位相空間の 2 つの選択肢が 1 つに圧縮され，位相空間の体積が半分になるという考え方である．したがって，この議論では，各論理操作あたり最小量の自由エネルギー $kT\log 2$ が必要になる*3．これまでに別のいくつかの提案があった．1 つの提案は計算操作の信頼性に焦点を当てたものであった．誤りの確率，たとえば q が含まれているので，最小エネルギーは $kT\log q$ であるという推測であった．

しかし，最近この問題は解決された．もしこの計算を注意深く，そして十分にゆっくりと実行しさえすれば，各操作あたりに必要なエネルギーは $kT\log q$ 未満で，$kT\log 2$ 未満であり，君たちが選ぶ任意の数未満である．理論上は，エネルギーの損失**なし**に計算を実行することが可能である．おそらく摩擦との類推を考えるとよくわかるだろう．実際にはいつも摩擦がある．典型的な現実のエンジンを調べると，いろいろな作動部分が互いに摩擦し合うので，いたるところで熱エネルギーが散逸することがわかるだろう．このエネルギー損失は普通は大きい．しかし物理学者はある型の理想化されたエンジン，いわゆるカルノーの熱機関の研究を非常に好む．この機関では，熱エネルギーが仕事に変換された後でその一部がまた熱にもどってきて，その際の運転での熱の最大利用効率を計算できる．このようなエンジンの動作は**可逆**循環過程である．つまり，ある特定の状態から出発して，1 サイクルの動作の後で，この状態にもどる．エネルギーを散逸せずにこの循環過程を実行することはできないが，たとえば摩擦による損失をできる限り小さくして，この機械が最大効率を達成するように運転することは理論的に可能であることを第 2 法則は保証している．不幸に

---

*2　この問題の歴史についての詳細な記述は，Zig-zag Path to Understanding, R. Landauer, *Proceedings of the Workshop on Physics and Computation Physcomp '94*，および Notes on the History of Reversible Computation, C. H. Bennett, *IBM J. Res. Dev.* 32(1), pp.16-23(1988)にある．［編者］

*3　これは現在のところ実際に実現可能な値よりはるかに小さな下限である．通常のトランジスターは各操作あたり 10^8kT のオーダーのエネルギーを散逸させる．［RPF］

も，このためには機械を無限にゆっくりと動作させねばならない！たとえば，君たちはすべてが熱平衡に保たれるようにエンジンの熱をまわりの熱浴に放出させたいかもしれないが，この機械をあまり速く運転すると円滑に放熱できず，熱はエンジンの各部分に伝わり，そこから散逸するだろう．しかし，肝心なことは，原理上はこのようなエンジンを作ることができ，物理学者がこのエンジンの研究から熱力学について多くを学んだということである．決定的な要請は可逆性である．

さて，コンピューターでも似た考え方がうまくいくことがわかっている．もし君のコンピューターが可逆ならば(可逆の意味についてはすぐ後で話す)，君が注意深くゆっくりと，規則上は無限にゆっくりと動かせば，エネルギー損失はいくらでも少なくできる．カルノーの熱機関の場合のように，あまり速く動作させるとエネルギーを散逸させるだろう．さて君たちは，なぜ私がこの問題を学問的問題と考えているかがわかるだろう．前に言ったように，現在のトランジスターはスイッチするたびに約 $10^8 kT$ を散逸するので，君たちはこの問題を少し間が抜けているとさえ思うかもしれない．しかし計算可能なものの限界についての議論の場合と同じように，このような問題は興味深い．大きさが原子程度の「トランジスター」を持つ，遠い未来の究極のコンピューターをわれわれが設計することになったとき，物理学の基本的法則がわれわれにどのような制約を加えるのかを知りたい．大きさを原子程度にまで小さくするとき，計算に伴うエネルギーを現実に問題にせねばならないが，このとき kT 以下で動作させられない理由は存在しないというのが答である．現在のトランジスターを部品としている最近のコンピューターのエネルギー散逸をどうすれば減らせるかといったような，もっと直接的な問題については，後で考察する．

5-2-1　可逆コンピューター



図 5-8　可逆計算

「可逆計算」の問題にもどろう．図 5-8 に一組の入出力線のあるブラックボックスとして描いた，次の特殊な種類の計算を考えよう．すべての入力線に対して 1 つ，かつただ 1 つの出力があり，この出力は入力によって決められると仮定する．(最もつまらない場合は，信号がこの箱の中を変化せずにただ通り抜ける場合である．) このような状況では，**出力**は**入力**以上の情報を持たない．つまり，入力を知っていれば出力が計算でき，さらに計算は「可逆」である．これは，AND(図 5-9)のような，通常の論理ゲートとは著しく対照的である．この場合には 2 本の線が入っているが 1 本の線しか出ていない．もし出力が 0 だとわかったら，入力は可能性のある 3 つの状態のどれかである．入力についての情報は復元不可能なように失われたので，この AND ゲートは不可逆である．そして OR ゲートも不可逆である(しかし NOT は不可逆ではない！)．言い換えると，エントロピーの不可避な減少を伴って，この入力の位相空間は出力の位相空間へと縮ん

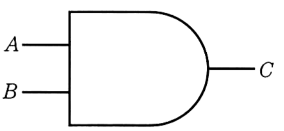


図 5-9　AND ゲート

でしまった．この減少はどこかでの熱生成によって補われているに違いない．コンピューターでのエネルギー散逸についてすべての人が犯していた過ちは，論理操作は，AND や OR のように，必ず不可逆であるという仮定にもとづいていた．ベネットたちが示したことは，論理操作は必ずしも不可逆ではないということであった．われわれの抽象的な「計算」では情報が増加しないという事実は，可逆計算にはエントロピーの損失が伴わないかもしれないことへの最初の手掛りである．これは実際に正しく，可逆コンピューターは最も効率のよい可逆機関のカルノーの機関のようなものである．われわれの抽象的な機械の運転によって生じる唯一のエントロピー損失は，機械が次の操作に対してリセットするときにだけ生じることがわかるだろう．

われわれは出力として，実際の計算結果に元の入力を加えたものを出力する，もっと直接的な意味で可逆的な「高度な」種類のコンピューターを考えることができる．つまり，このコンピューターは出力データが印刷されたテープ(たとえば)に入力データも印刷する．これが計算を可逆にする最も直接的な方法である．そのような計算は，理論上はエネルギーを費やさずに実行できることを後で示す．エネルギー損失は機械を再出発させるためにリセットするときにだけ生じ，面白いことにこの損失は計算自身の複雑さではなく答のビット数だけで決まる．機械の中にぶんぶん回っている何十億の部品があったとしても，もし取り出した答が 1 ビットだけなら，$kT \log 2$ が動作に必要な全エネルギーである．

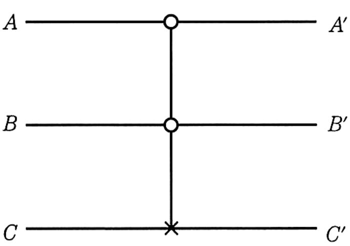


図 5-10　CCN ゲート

われわれはこのコースの初期に，いくつかの可逆ゲートについて勉強した．私が述べたように，NOT は可逆ゲートの 1 つである．われわれが調べたもっと複雑な一例は，フレトキンの二重制御付 NOT ゲート(CCN ゲート)であった(図 5-10)．このゲートでは，線 A, B は制御線であり，両者が 1 でなければ C はそのままであり，両者が 1 であれば C は NOT C になる．このゲートは，出力を別の CCN ゲートに通すことによって入力データが再び得られるという意味で，可逆である(2-3 節参照)．

さて，いくつかの可逆計算をもっと詳しく考察し，最小エネルギーの必要がないことを証明したい．君たちがふつうは計算と考えないかもしれない計算，つまり複写作業(3-5 節のチューリングの複写機の議論を思い出すこと)から始めよう．これは新しいことは出てこないので，頭の悪い計算のように思えるが，しかしこれはエネルギー散逸の問題の根底にあるいくつかの考え方への有効な導入になる．エネルギーを少しも消費せずにある場所から別の場所へ情報を複写できることは，原理的にさえ，決して明らかなことではない．ところで，なぜこの操作にエネルギーが全然いらないのかを示すのは容易である．われわれは一組のデータとそのコピーを，テープ上の 2 つのメッセージ

として考えることができる．両者は同一のものである．われわれはオリジナルのメッセージが何であるかを知っているか，知らないかのどちらかである．最初の場合には，テープを消去するのに自由エネルギーは費やされないし，またコピーのテープを消去するのにもエネルギーは費やされない．前に議論したように，必要なときにビットをひっくり返すだけでよい．第2番目の場合には，オリジナルのテープを消去することによって自由エネルギーは消費されるが，コピーのテープについては自由エネルギーは消費されない．われわれは最初のテープが示す内容を知って，この情報を使ってここでもビットをひっくり返してコピーのテープを消去できる．簡単に言うと，この(データとコピーの)組にはデータの組だけより多くの情報は含まれていないので，消去する際に，第2番目の場合よりも最初の場合の方が多くの自由エネルギーを必要とするはずはない．このような議論の仕方は可逆計算の世界でよく見かける型の議論である．

5-2-2　複写計算

このような考え方をもう少し具体的にしよう．少しあとで自然界に見出される複写機，すなわち生物細胞に見出されるRNA分子を調べる．しかしまず最初に，人工複写機の2つの例を考察したい．この議論はベネットによるものである．

ある非常に一般的な複写処理から始めよう．われわれは原物(オリジナルの物体)を持っていて，これを模型と呼ぶ．これは何らかの方法で1つの0または1つの1を保持できる．これは2つの安定な状態を持つ(双安定な)ある種の物理的装置である．われわれはもう1つの物体が必要で，これを複写機と呼ぶ．これも1つの0または1つの1を保持できる．双安定な装置の一例は図5-11のようなポテンシャルの井戸によってモデル化されうるものである．私は物理的に実現可能な実例を次節に示す．この抽象的な図の意味は，装置の黒丸で表されたある部分が2つの安定な状態のうちのどちらかに存在しうるということである．ここで，左または右の窪みは，たとえば，1または0を意味している．この曲線は装置の中での黒丸の位置によるポテンシャルエネルギーを表している．2つの窪みはこのエネルギーが最小のところであり，黒丸が好んでいる場所である．2つの窪みは同じ深さなので，黒丸によって最初に占有される可能性は等しい．この操作を考えるために役立つ方法は，黒丸をボールとし，曲線をこのボールを制約する実際の形にすることである．エネルギーをボールに注入すると，このボールは窪みの両側の斜面を上下に動く．ボールに十分なエネルギーを注入すればボールは丘を登り隣の窪みに入るが，これはわれわれの模型でビットの状態が変わるのと等価である．丘の高さは，遷移が起こるのに必要なエネルギー量で，障壁ポテンシャルと呼ばれている．実際の操作では，装置を安定に保つために，黒丸が表しているものが何であっても，その熱による普通のゆらぎは障壁ポテンシャルよ

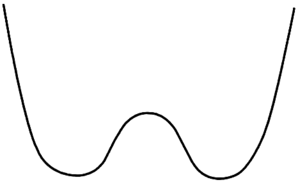
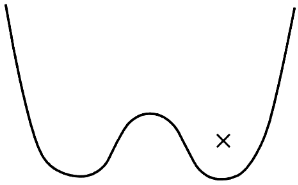


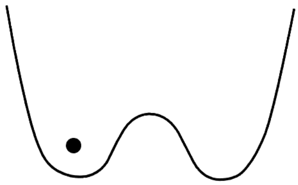
図5-11　ポテンシャルの井戸

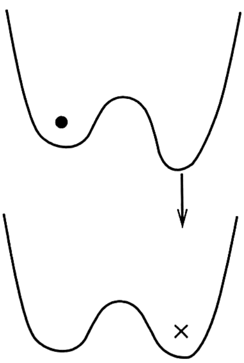
りずっと小さくなっていてほしい．これを視覚化する別の方法は，仕切りで半分に分離された1つの箱の中に黒丸があると想像することである．この障壁ポテンシャルはこの黒丸が一方の半分からもう一方の半分へ移るのに必要なエネルギーである．

模型と複写機の両方がそのようなポテンシャルによってモデル化され，模型はある状態にあると仮定しよう．その状態はランダムであってよく，われわれはそれが何かを知る必要はないが，説明のために，たとえば図5-12に示されているようなものだとしよう(ここで模型の場合は黒丸ではなく×印を使った)．この複写機はどのように動作し始めるだろうか？　複写機はある標準状態にあるに違いない．ランダムな状態にあるはずはない．なぜならば複写操作は複写機をある決まった状態にすることを意味し，これを行なうためにわれわれは仕事をしなければならないからである(箱と仕切りという類推を使うと，圧縮しなければならない)．代わりの方法として，複写前の「模型-複写機」の可能性のある選択肢の数(もしこの模型がランダムにセットされると4つ)と複写後の数(2つ)とを比較して，位相空間の考察を利用できる．したがって，複写は論理的に不可逆な過程だろう．たとえば，複写機が模型とは逆の状態から出発するとしよう(図5-13)．明らかに，複写は何らかの手段で黒丸を一方の窪みからもう一方の窪みへ移すことを意味している．そのためにポテンシャル曲線を操作して，もう一方の窪みが黒丸にとってエネルギー的に有利であるようにすることが可能でなければならない．複写機には，障壁の高さと2つの窪みの相対的な深さという，2つの調節可能なパラメータがあると仮定する．さらに，窪みの深さは複写機と模型との間に働くある力によって変えられると仮定する．(これらの仮定がとてもごたごたしていて抽象的だと感じても心配することはない！　すべては明らかになるだろう．) この力の効果は図を傾けることと同じであるため，これを「傾き」の力と呼ぼう．われわれはこれらの2つの操作を組み合わせて複写機の黒丸を移動させるが，重要なことは，この黒丸が到達可能な最小値が常に1つだけ存在するように2つの操作を組み合わせることである．

われわれが行なうのは次のようなことである．われわれは模型を複写機からいくらか離して開始する．ある距離だけ離していても模型は複写機にわずかな傾きの力を及ぼす．この力によって模型で占められている方の窪み(×印)に対応する複写機の窪みの深さが増加するようにする．したがって，複写機のポテンシャルはまず，図5-14に示されているように，少し変形する．複写の第1段階では複写機のポテンシャル障壁をゆっくりと低くする．こうすることによって黒丸が位置を変えることに対する障害が取り除かれ，この黒丸はもう一方のビットの状態へさまよっていくことができるようになる．何が黒丸をさまよわせるのだろうか？　模型からの「傾き」の力が登場するのがここである．第2段階では，われわれは模型を複写機にゆっくりと近づけ


図5-12　模型の初期状態


図5-13　複写機の初期状態


図5-14　複写機の初期の変形

る．近づけるのにつれて傾きの力は増加する．この力は複写機のポテンシャルをゆっくりと歪ませていき，図5-15に示されているように行き先の窪みのエネルギーを低下させる．そこで黒丸はポテンシャル曲線をなめらかに滑り降り，新しいエネルギー的に有利な窪みを占める．第3段階では，ポテンシャル障壁を元にもどして黒丸が新しい位置に確実にいるようにし，最後の第4段階では模型を遠ざけて，複写機のポテンシャルを正規の状態に復元する(図5-16)．これがこの複写機の基本的な概念である．

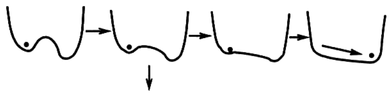


図5-15 ポテンシャル障壁の低下と傾き

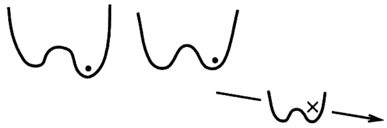


図5-16 系の最終状態

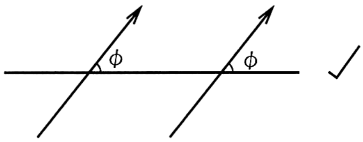
この考えをもっと楽しむことができる．たとえば，適切な物理系に対して，第1段階でポテンシャル障壁を下げている間に，(もし気になるのなら)黒丸をしっかりと保持するように，模型を複写機に近づけて傾きの力で黒丸の入っている状態を低くすることが考えられる．それから，逆の傾きを与えるために模型を反対側に移動させる．これは1つの変形であるが，基本的な概念を著しくは変えない．

この処理で決定的に重要なことは，ゆっくりそして注意深く実行する必要があるということである．飛躍や突然の変化があってはならない．黒丸を1つの窪みからもう1つの窪みへ移す最もやさしい方法は，模型を迅速に近づけて，窪みを望みどおりに偏らせ，それからポテンシャル障壁を切り取ることだろう．黒丸は新しい窪みに飛び込むが，この全過程は，とても迅速である限り，現実の系におけるエネルギー散逸を常に伴う．しかし，この処理が十分に「おしとやか」であれば，障壁を低下させ，窪みを傾け，そして複写する諸操作はエネルギー散逸無しに実行できる．その理由は，基本的には，新しい状態へ移動する黒丸の運動エネルギーや，障壁を上下させるときになされる仕事のような，エネルギーの散逸に寄与する物理量がそのような状況のもとでは無視できるためである．ところで，もし模型がどの状態にあるのかを知らなくても，この処理はうまくいくことが君たちにわかるはずである．

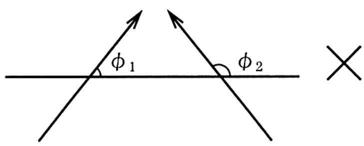
ベネットがこれらのことをすべて発見したとき，このようなことが発見できるとは誰も考えなかった．しかし，多くの予備的な基礎研究が彼のIBMでの同僚ランダウアーによって1961年にすでに行なわれていた．ベネットの研究は，反論されねばならなかった多くの偏見に囲まれていた．私は彼の議論に何らの間違いを見出さない．私はカルテクのミードにコンピューターのエネルギー消費を調べるよう依頼されたときに，これらの論文の全部を読んで徐々に最小エネルギーが存在しないと結論するようになった．これは私にとって驚きであった！ベネットの結果が発見されたのはその4年前であったが，その頃まだ彼の研究について論争をしている人たちがいた．この種の問題を君たちが自分で解くことは楽しいことである．というのは，第1章で私が言ったように，OK，君たちがこの問題を最初に解くわけではないが，君たちは少なくともその問題を理解するからである！

5-2-3　物理的な実行

前の例にもどり，本質的にこの例を物理的に実現しているものを示そう．あれこれ考えるのも楽しい！　われわれはある種の双安定な物理的装置が必要であるが，そのような装置は存在する．2つの羅針盤の磁針，つまり旋回軸のある2つの磁気双極子がそれである．磁針の一端はN極で他端はS極であり，だれでも知っているようにN極はS極を引き付け，S極はN極を引き付ける．それ以外の場合は反発する．さて，模型と複写機の両方ともこのような磁針の対で構成されていると仮定する．解析しやすくするために，対の磁針は同じ方向を向くように連結されているものとする．これは各系が磁針と水平線とのなす角度ϕという1変数だけで解析できることを意味している．そのため図5-17に示すような，許される状態と許されない状態がある．許されない場合は，どのような場合でも明らかに不安定である．



(a)許される角配位



(b)許されない角配位

図5-17

さて，対になっている2つの磁針のいろいろな向きの全部が同じポテンシャルエネルギーを持っているわけではない．これは図5-18に示されている2つの状態を比べると明らかである．一方の磁針の先端がもう一方の磁針の頭部を引き付けている最初の対は，明らかに安定である．2つの磁針が鉛直方向を向いていて，2つのN極が反発しあう2番目の対は不安定で，2つの磁針は最初の対の状態またはその鏡像の状態になろうとする．われわれは角度ϕの状態のポテンシャルエネルギーを実際に計算できる．これは近似的に(われわれには十分な精度だが)次のように表される．

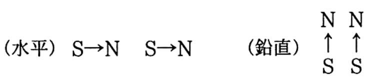


図5-18　安定な状態と不安定な状態

$$\text{ポテンシャルエネルギー} \approx \sin^2\phi \qquad (5.17)$$

このポテンシャルエネルギー関数は図5-19のグラフのようになる．これがわれわれの抽象的なポテンシャルの井戸に良く似ていることを注意してほしい．最小値は$\phi=0$および$\phi=\pi$にあり，これは磁針が「水平」で安定な2つの状態に対応し，一方，最大値は$\pi/2$および$3\pi/2$での磁針が鉛直な2つの状態に対応している．(このグラフは丸まっていて$\phi=0$と2πは同一の点であることを思い出すこと．）この系は明らかに双安定で，磁針がいったん2つの最小値のうちの1つになれば，磁針をもう1つの最小値へ押しやるためにエネルギーを費やさねばならないことがわかる．

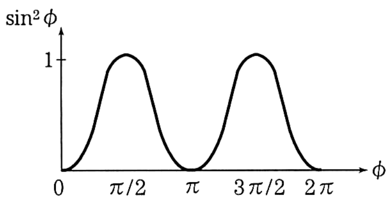


図5-19　ϕの関数としてのポテンシャルエネルギー

この場合に障壁を操作するために，鉛直な磁場Bを導入する．この磁場はポテンシャルエネルギーに項

$$-B\sin\phi \qquad (5.18)$$

を付け加えることがわかる．Bを増加させると，図5-20に示されているように，0とπの状態間の障壁を低くするという効果がある．(この正確な効果を評価するためにいろいろな場合を調べてみることができる．）傾きの力は，前のように，模型を複写機に近づけた結果として生じる．この例では，模型の何がこの力を引き起こしたかがわかる──この力は模型の磁針からの磁気力である．この力はBに垂直

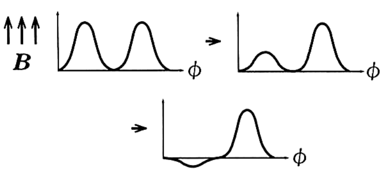


図5-20　双極子複写機の障壁操作

で，模型の磁針の方向を向いている．これを b と呼ぶと，これはポテンシャルエネルギーに

$$-b\cos\phi \tag{5.19}$$

だけ寄与する．これは明らかに $\pi/2$ と $3\pi/2$ についての対称性を取り除き，傾きを表す．これでわれわれは複写処理がどのように行なわれるかがわかる．複写機は開始時には標準状態にあるとする．$\phi=0$ の状態(→→)を標準状態とする．ゆっくりと磁場 B をかけていき，あるいはその代わりに複写機を B の弱い領域から B の強い領域へゆっくり移動させて，障壁を取り除く．この段階では，双極子は鉛直である(図 5-21)．次に模型を近づける．この模型は複写機の対をこれまでもわずかにかき乱してきたが，これまでは目立った効果を与える強さではなかった．模型を近づけていくと，模型の作る磁場は複写機の磁針を傾かせひっくり返して新しい状態にするが，突然にではない！(これは新しい状態が適切になっている場合である．標準状態と模型の状態が一致している場合は，磁針は元の位置にもどるだけである．)模型を移動し，障壁を復活させるために複写機を磁場 B の中から取り出すと複写は終了する．

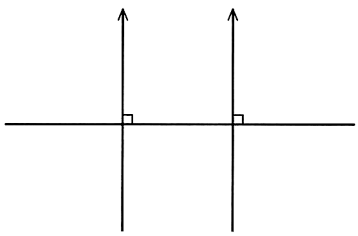


図 5-21　初期の(不安定な)複写機の状態

模型の状態を知らなくても，この複写法がうまくいくことを確かめられる．もしゆっくりと実行すれば，この操作ではエネルギーは費やされず，電流も流れず，何も発生しないことを理解するのは難しくない．私の以前の議論は原理を示すことが目的であったが，この実例の方がおそらく理解しやすいだろう．

5-2-4　生体コンピューター

いま示した 2 つの双極子の例には確かな物理的基礎があるが，不自然だということは否定できない．しかし，自然界では実際に複写処理が行なわれている．この処理には力学的な力ではなく熱力学的な力が関係している*4．この処理は生体細胞の中でのタンパク質の合成の一段階として起こる．さて，君たちはタンパク質とは何かをおそらく知っているだろう．(トリプトファンやアラニンのような)アミノ酸の長くてねじれた鎖状分子である．君たちはこれらが生体の構造と機能の両方にとっていかに中核的なものであるかを知っているかもしれない．しかし，これらの製造という複雑な仕事を適切に理解するには生化学の理解が必要であるが，本コースの範囲を越えているので，私はそのような話はしない．ただ私の念頭にあるこの複写「機械」がどのように振る舞うかを君たちが理解するための十分な背景を与えたい．

生物は一般に莫大な種類のタンパク質を含んでいて，各タンパク質は特定のアミノ酸のある組合せによって一義的に定義されている．もし細胞がこれらの分子を製造するのであれば，明らかに各々の型のタンパク質に対する一組の「設計規則」がどこかで利用可能になってい

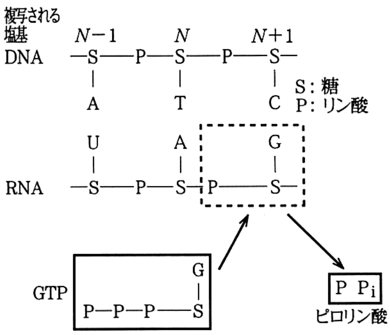
---

*4　この話題の議論に関する文献としては，C. Bennett, *Int. J. Theor. Phys.* 21, pp. 905-940[1982]を参照せよ．[RPF]

るはずである．この情報は，有名な「二重らせん」構造を持ち細胞核の中にあるDNA(デオキシリボ核酸)分子の中に存在する．DNAは二重の鎖で構成され，その各々の鎖は交互に並んでいるリン酸とペントース糖類から作られている．各々の糖類には，A(アデニン)，T(チミン)，C(シトシン)とG(グアニン)の4塩基のうちの1つが付いている(塩基-糖-リン酸のグループはヌクレオチドと呼ばれている)．タンパク質合成のための符号を与えるのはある塩基配列である．

タンパク質合成は2段階に分けられる．われわれが興味を持つのは第1段階だけであるが，この段階では，メッセンジャーRNA(またはmRNA)と呼ばれている，別の塩基が付いた糖リン酸の線状の鎖が形成されることが必要である．DNA上の符号は(後で話題にする厳密な対応規則に従って)このRNAの鎖へ塩基ごとに複写され，複写が終わるとmRNAは核から離れ，タンパク質を作るのを手伝うために他の場所へ動いていく．複写を実行する機械はRNAポリメラーゼと呼ばれる酵素である．次のようなことが起こる．DNAと酵素は，ATP(アデノシン三リン酸)，CTP, GTPおよびUTP(Uは別の塩基のウラシル)のような多くの三リン酸を含んだ風変わりな生物学的なスープの中で漂っている．これらは本質的に2つの余分なリン酸がくっついたヌクレオチドである．このポリメラーゼは複製を作ることになっているDNAの鎖の1つのうちのどこかの部分に付き，鎖に沿って動き，スープの中に存在する4つのヌクレオチドのうちの1つと成長しているRNAの鎖とを反応させ，塩基ごとにRNA複製を作り上げていく．(ここでの決定的な条件はRNAが4つの塩基A, G, CおよびU(Tではない)から作られていることと，RNAの鎖がDNAの鎖と相補的でなければならないことである．この相補関係はDNA上のAはRNA上のUと，TはAと，CはGと，そしてGはCと組まねばならないという関係である)．ヌクレオチドは三リン酸という形で供給され，くっつく過程の間に2つのリン酸はこのスープにもどされるが，(ピロリン酸として)結合したままになっている．選択されたヌクレオチドは適切なものでなければならない．つまり複写されようとしているDNA鎖上の塩基と相補的でなければならない．たとえば，DNAの鎖に沿って動いている酵素が塩基Cに行き当たったとする．処理のこの段階ではポリメラーゼとスープのGTP分子との結合がポリメラーゼとUTPやATPとの結合よりエネルギー的に有利になっている．つまり相補性はエネルギー的な条件によって強制される．その結果，RNAポリメラーゼは優先的にGTP分子を選び，ピロリン酸をスープにもどし，DNAに沿って動き，次の相補的なヌクレオチドを探す．

図式的には，図5-22のようになる．



RNA(N番目の塩基)+(G-S-P-P-P)
→RNA(N+1番目の塩基)+P_i-P (ピロリン酸)

図5-22　mRNAの形成

この生化学的な過程での酵素の役割は触媒としてである．つまり，

これらの酵素は反応速度に影響を与えるが，反応が進む方向には影響を与えない．化学反応は可逆であり，ポリメラーゼの反応が逆の方向に行く，つまり酵素が作業している mRNA の鎖を元にもどすことも可能である．そのような場合には，酵素は周囲のスープからピロリン酸を取り出して mRNA の塩基に付け，三リン酸の 1 つにして周囲へ放出する．この酵素は逆の方向に動き，G を食べて，動き，C を食べて，動き，…，すべてを元にもどしてしまう*5．反応がどちらの方向に進むかはスープの中のピロリン酸と三リン酸の相対濃度によって決まる．もし ATP, GTP などが多くあり，遊離したピロリン酸がそれほど多くなければ，酵素は mRNA ヌクレオチドを引き離すためのピロリン酸をあまり見つけられないので，酵素が反応を逆もどりさせる率は低下する．他方，もし遊離したピロリン酸が三リン酸よりも過剰に存在すれば，この反応は逆の方向に行く傾向になり，複写を元にもどし，この複写をだめにしてしまうだろう．

これらの相対濃度を，系が計算中のある時点で利用可能な状態の数と解釈できる．もし周囲に多量の三リン酸があれば，利用可能な(入っていける)状態は前進方向には多く存在し，後退方向には少ない．したがって，RNA ポリメラーゼは前進方向にある状態に入る傾向があり，この過程でそのエントロピーは下がる．濃度の差によって決まる自由エネルギーの差が進行方向を決める．この濃度をちょうどピッタリにすると複写機は永久に振動し，コピーを入手することはできない．実際の細胞では，ピロリン酸の濃度は加水分解によって低く保たれ，複写処理のみが起こり，その逆処理は起こらないようになっている．全 RNA ポリメラーゼ系はエネルギー利用という点では特に効率的というわけではなく，ビットあたりおよそ $100kT$ を散逸させる．酵素がもう少しゆっくり動くとエネルギーの浪費は減るのだが，生きるためにはある程度の速さが必要である！(もちろん，この反応速度は濃度勾配によって変化する)．それでもなお，ビットあたり $100kT$ というのは典型的なトランジスターの場合の 10^8kT に比べるとかなり効率がよい！

繰り返すと，この節で学んだ教訓は複写に必要なエネルギーには絶対的な最小量がないということである．しかし，ある速さで複写したければ，ある限界が**存在する**．

5-3 計算：エネルギーコスト対速さ

速さの問題は重要なので，ある有限な時間に計算を実行するのに必要な自由エネルギーの量に対する式を書き下したい．この式は少なくともわれわれの議論をもう少し実際的なものにする．現在のところコ

---

*5 この機械の動作は本質的にランダムであるが，それでも外部から課された条件でこの処理がある方向にゆっくりと進む様子をとらえて，ベネットはこのような機械を「ブラウン型コンピューター」とうまく名づけた．[RPF]

ンピューターの世界では可逆計算の可能性はほとんどない．しかし現在の 10^8kT から kT 以下にするよりは，もっと早く役立ちそうな応用を予見できる．(不可逆的に $2\sim3kT$ にすることはできるが，これ以下にはできない．) たとえば，何千というプロセッサーがいっしょに働いている並列処理のアーキテクチャでの誤り発生の問題を調べることができる．このような状況での符号化による誤り訂正が問題になっているが未解決である．ことによると機械の中の装置はすべて可逆にできるかもしれず，そうすると誤りが起こったときに気づくことができるかもしれないという考えが浮ぶ．そのような可逆装置のエネルギーコストはどれくらいになるだろうか？　可逆装置は近いうちに応用されるかもしれない．このことはこの議論を君たちにとってより実際的なものにし，計算は工学なので，君たちはこのことを高く評価するかもしれない！　ともかく，私は遠い将来への自分のとっぴな学問的関心についてこれ以上弁明すべきではない．

可逆計算の例としてわれわれが示したのはDNAを複写する化学過程であった．これはときどき思い出したように進む(君たちがそう呼びたければ)機械である．少し前進し，それから後退するが，ある駆動力のために後退より多く前進するので，結果的にある計算をする(この場合は複写をする)．これをもっと一般的な考察に対するモデルとして取り上げて，そのような過程におけるエネルギー散逸の式を導出するためにこの「ブラウン」的概念を使おう．この式は計算の間のエネルギー散逸の一般式ではないだろうが，その算出にどのように取り掛かればよいかを君たちに示すはずである．しかし，この議論の前にその一般式[*6]を示しておこう．そうすれば，その後で述べることを実例として見ることができる．

可逆コンピューターがあると仮定しよう．ふつう，これを可逆的に運転するときに，つまり過程が無限にゆっくりであるときに費やされる自由エネルギーはゼロであるが，われわれはこのコンピューターを r の割合で前に駆動すると仮定しよう．言い換えれば，すべての段階で，計算の前進ステップ数は後退ステップ数に比べて r 倍であるとする．このとき，この過程で計算の各ステップあたりに費やされねばならない最小エネルギーは

$$kT\log r \tag{5.20}$$

だというのが一般的結果である．r が小さくなるにつれて，エネルギーが小さくなることに注意すること．

ブラウン型コンピューターを考察することによってこの規則を説明しよう．固有のエネルギーを持つ特定の状態にある系，すなわち装置を想像してほしい．このコンピューターは前進または後退して新しい

＊6　この規則はかなり一般的だが，例外があり，少し訂正が必要である．そのような一例の「弾道」コンピューターについては5-5節で議論する．

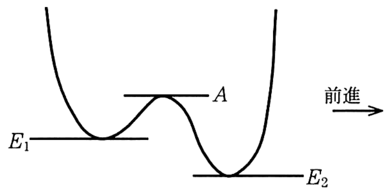


図 5-23　一般的な遷移

状態にいくことができ，各遷移は計算の実行(前進)または取り消し(後退)のどちらかに対応している．われわれは図 5-23 のエネルギー準位の図を使ってこの状況をモデル化できる．われわれのコンピューターはエネルギー E_1 または E_2 の 2 つの状態のうちの 1 つにあると仮定する．2 つのエネルギーは一般には等しくない．さて，この装置は E_1 から E_2 へ進行することもできるし(前進ステップ)――エネルギーは計算の方向に向かって低いので前進ステップとよぶ――，あるいは E_2 から E_1 へ進行することもできる(後退ステップ)．2 つの状態のエネルギーは等しくなるかもしれないが，そのうちの 1 つは外部から駆動力をかけることによって実質的に低くできる．われわれはこの図に「活性化エネルギー」A を導入したが，これは系にどのような種類の遷移を引き起こすためにも供給せねばならないエネルギーである．さて，ゆらぎのエネルギーが A より大きいときにはいつでもコンピューターが E_1, E_2 の状態間を移動するようにさせる，まったくランダムな熱ゆらぎの効果に焦点を当てよう．そのようなゆらぎは装置をどちらの方向へも行かせることができる．われわれは装置が各方向へ行く速さを計算できる．これらは等しくないだろう．おおざっぱに言うと，系がエネルギー E_i の状態に行く確率は，障壁(つまり A)を越えて E_i に入るのに十分なエネルギーを偶然に得る確率である．E_1 から E_2 への前進ステップに必要なエネルギーは明らかに $A-E_1$ であり，他方 E_2 から E_1 へ行くのに必要なエネルギーは $A-E_2$ である．ある状態から(正の)エネルギー δE だけ異なる他の状態へ遷移する確率は，統計力学の標準的な結果によれば，

$$C\exp(-\delta E/kT) \qquad (5.21)$$

である．ここで C はこの環境の中での熱ゆらぎの情報を含んでいる因子である．この因子は状態間のアンサンブル遷移確率を調べることにより，位相空間(エントロピー型)の解析を通して計算できる．われわれは状態間の遷移速度(率)に関心があるが，これは(5.21)に類似した式によって記述できる．われわれは単に，たとえば X のような別の因子を挿入すればよいので，それらは

$$\text{前進速度} = CX\exp[-(A-E_1)/kT] \qquad (5.22)$$

および

$$\text{後退速度} = CX\exp[-(A-E_2)/kT] \qquad (5.23)$$

と表される．因子 X は問題にしている物質の分子のいろいろな性質(平均自由行程，速度など)に依存するが，われわれが関心を持つ性質はこの因子が E に依存しないという性質である($E_1=E_2$ の場合の遷移速度を考えよ)．したがって，前進速度(率)の後退速度(率)に対する比は次のように書ける．

$$\exp[(E_1-E_2)/kT] \qquad (5.24)$$

これは引きつづいて生じる状態のエネルギー差だけで決まる．この式は，この計算(＝反応)が進む速度，およびこれを駆動するために要求される各ステップ間のエネルギー差についてある見通しを与える．エ

ネルギー差E_1-E_2が大きくなれば，この機械はE_1からE_2へとより迅速に飛び越せるようになり，計算(＝反応)がより速くなる．

われわれは，この速度の比(5.24)がrに等しいと置くことによって，この結果を以前の一般式と結びつけることができる．すなわち，ステップあたりに費やされるエネルギーに対して，

$$kT\log r = E_1-E_2 \tag{5.25}$$

が得られるが，これは意味をなしている．

コンピューターを特定の方向に駆動することについてもう1つの例を示そう．今度はエネルギーが異なるのではなく利用可能性が異なる計算の状態を考察する．つまり，このコンピューターは，その状態のエネルギーではなく，それが入ることのできるある種の等価な状態の数にもとづいて，次にどの状態へ入るかを選択する．DNA複写機にそのような状況の例がある．そこでの計算のステップは，RNA酵素が塩基をRNAの鎖に付けてピロリン酸を周囲へ遊離させることであった．その逆のステップは溶液からリン酸を取り出して塩基をちぎり取ることであった．各ステップはエネルギー的に等価であるが，スープの中の化学物質の相対濃度によって，一方のステップが他方のステップに比べて有利になる．リン酸は欠乏しているが利用可能な塩基はたくさんあると仮定しよう．このときRNAの鎖に塩基がくっついてリン酸が解離する系の(前進)状態(そのような状態のすべてを等価と考える)の数は，リン酸が引ったくられて塩基が解放される系の状態(ここでもそのような状態はすべて同じと考える)の数よりも多い．そのため，利用可能性の低い状態ではなく，利用可能性の高い状態の方へ移動する確率が大きいという意味で，拡散によって前進するように設計されたコンピューターを想像できる．図式的には，図5-24に示された状況になる．ここでn_iは利用可能な状態の数である．この状況で後退速度に対する前進速度の比は(少し考える必要があるが)次のようであることが示せる．

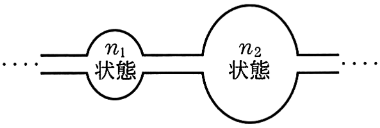


図5-24　状態の利用可能性

$$r = n_2/n_1 \tag{5.26}$$

以前にわれわれは，ある系のある配位のエントロピーを次のように定義した．

$$S \approx k\log W \tag{5.27}$$

ここで，Wはこの系がその配位にあることが見出される確率である．したがって，次のような式を書ける．

$$kT\log r = kT(\log n_2-\log n_1) = (S_2-S_1)T \tag{5.28}$$

(いろいろな定数因子が相殺してこの等式が残る)．言い換えれば，この過程のステップあたりのエネルギー損失は，いつもの温度因子を別にすると，そのステップで生成されるエントロピーに等しい．これも意味をなす．

そこでこの場合には，一般式がいま得られた式に変形されたことが

わかる．さて，「現実世界の状況で，計算のステップあたりに使われるエネルギーを最小にできるだろうか？」という興味深い問題が生じる．もし効率的な可逆コンピューターが存在すれば，前進と後退の可能性は等しいことをわれわれは知っており，エネルギー損失はない．これに対して支払う代償は，計算に無限の時間がかかるということである．われわれにはいつ終わるのか永久にわからない．そのため前に述べたように，コンピューターを前進させるために，われわれは引っ張ったり，前進していく先のエネルギーを下げたり，利用可能な状態を増やしたり，その他いろいろなことをしたい．前進速度 f が後退速度 b より少しだけ大きいので計算はやっと進むと仮定しよう．そこで，

$$f = b+\Theta \tag{5.29}$$

と書こう．ここで Θ は小さい．Θ が小さい場合のわれわれの一般式は

$$\begin{aligned}\text{ステップあたりのエネルギー} &= kT\log[1+(\Theta/b)] \\ &\approx kT\Theta/b = kT(f-b)/b \end{aligned} \tag{5.30}$$

になる．数学的に不正確ではあるが，物理的にうまい解釈をこの式に与えることができる．この公式を，それとほとんど等しい公式

$$\text{ステップあたりのエネルギー} = kT\frac{f-b}{(f+b)/2} \tag{5.31}$$

で置き換える．この公式はもとの公式とオーダー Θ^2 の項だけ異なっている．この分数の分子は前進して計算を行なう速さである．コンピューターが計算をさまよいながら行なう速度を表すという意味で，これは1秒あたりのステップ数で測定した速度に少し似ている．分母は遷移を行なう平均速度，つまりコンピューターが後退したり前進したりという振動を行なう度合いを表す．これはコンピューターがもどることなく一方向へ一連の処理を行なう場合の速度の，後退または前進できる最高速度として大まかに解釈できる．つまりこれは最大の可能なドリフト速度である．そこで，われわれは近似的に次のように書くことができる．

$$\text{ステップあたりのエネルギー損失} = kT\frac{v_{\mathrm{drift}}}{v_{\max}} \tag{5.32}$$

この代わりに，変数として時間を強調し，次のように書くこともできる．

$$\text{エネルギー損失/ステップ} = kT\frac{\text{最小所要時間/ステップ}}{\text{時間/実際のステップ数}} \tag{5.33}$$

次に可逆計算におけるもっと一般的な問題を考察しよう．

5-4　一般的な可逆コンピューター

これまでに繰り返し述べてきたように，もし計算が可逆ならば，後で逆もどりするために使えるように，普通は失ったり廃棄したりする

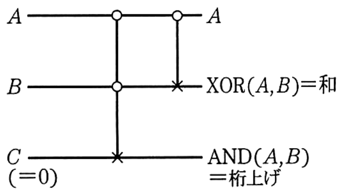


図 5-25　可逆ゲートによる2ビット加算器

多くの情報を保存せねばならない．そのような機械の論理ゲートは，われわれがほしい論理計算への答だけではなく，多くの余計なビットを出力する．実際のゲートの場合の簡単な実例は，可逆ゲートから組み立てられた簡単な加算器である．2-3節で私は君たちに可逆ゲート，特に CN と CCN ゲートを使って(あるいは，CCN ゲートだけでよい．これから他のすべてのものが組み立てられる) 3ビットの全加算器を組み立てる問題を出した．もっと簡単な例として，2ビットの半加算器は図5-25のように組み立てられる．C 入力はゼロにセットされたままである(3ビットの全加算器には，ゼロにセットされたままの4本目の入力線を加える必要がある)．$A+B$ の和と桁上げのほかに，A 線がこのゲートを通り抜けている．出力から入力(A,B)を再構成できるためには，このビットが必要なことがわかる．3ビット全加算器を調べると，出力に2つの余分なビットがあることがわかるだろう．このように，一般には，過去の論理演算の歴史を思い出すためのある量の不要品が常に必要である．可逆ゲートについての主な制約を以下のようにまとめることができる．コンピューターを前進動作させるとき，先のステップに曖昧さがあってはならないことは明らかである．もし「goto」が入力されたら，どこに行くのかを知らねばならない．可逆機械では，**後退**ステップにも曖昧さがあってはならない．どこにもどることになっているのかがわからないという状況になってはならない．可逆計算と普通の不可逆計算とを根本的に異なったものにしているのは，この特徴である．

ベネットがそうしたように，最も一般的な計算過程を考える．そうすると，可逆計算の主唱者に向けられた批判に答えることもできる．たがいに連結された(可逆)論理装置の系があるとして，これにあるデータを入力する．われわれは一組の「標準の」0も入力せねばならない．これは可逆ゲートを制御するために特定の値にセットされている一組のビットである．(もしある0を「標準の」1にしたければ，0の1つのNOTをとればよい．もちろんこれは可逆である！) 各論理装置はそれぞれの仕事をし，──ガー，ガー，ガー──最後に出力が出てくる．これはほしい答と歴史テープというごみの山の両方である．この様子は図5-26に示されている．さて，この図は計算が空白のテープ(または現在のテープ)から開始して，無秩序状態で終わっているように見せている．そこで当然，誰もがエントロピーはどこに行くのかと言った．すなわち「0のランダム化は(ベネットの描像によれば)機械の運転に燃料を供給している．データの保存がどのようにして計算を実際に可逆にするのだろうか？　これはすべての熱が入っていった水を投げ捨てないで保存すると不可逆熱機関を可逆にできると主張するのに似ている．もし水を投げ捨てなければ，確かに君たちは系の歴史について必要なすべての情報を持つが，それは水分子の動きを逆にして，エンジンを逆に運転できるようにすることを意味しない！」と誰もが言った．熱力学の場合には，たしかにこれはばかげている．し

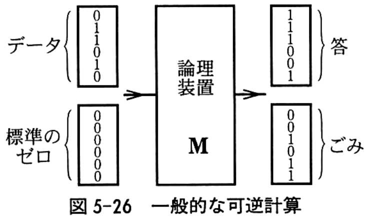


図 5-26　一般的な可逆計算

かし計算の場合にはそうでない．この系にもう1つのテープを加え，そして結果を別の機械へ入力することによって，この困難を回避できる(図 5-27).

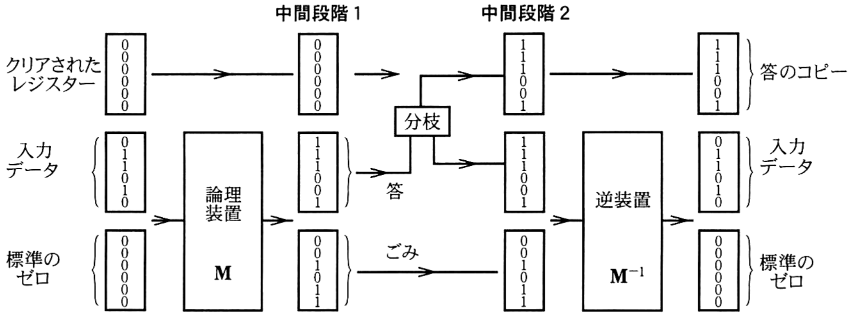


図 5-27 エントロピー損失ゼロの可逆コンピューター

この図の意味を説明してみよう！ われわれが加えた新しい論理装置は元の装置の逆であり(したがって，これに $\mathbf{M}^{-1}$ というラベルを付けた)，この装置も可逆である．$\mathbf{M}^{-1}$ に $\mathbf{M}$ の出力を送ると，$\mathbf{M}$ が行なったすべての仕事を元にもどし，$\mathbf{M}$ への入力が $\mathbf{M}^{-1}$ から出力される．新しいテープはわれわれの計算の答を複写するために使うクリアされたレジスターである．さて，前と同じよう，入力データと制御のための標準の0を $\mathbf{M}$ に入れることから始める．$\mathbf{M}$ は出力と歴史テープ(この図には「ごみ」と記されている)を出す．歴史テープは直接 $\mathbf{M}^{-1}$ へ入力される．またデータの出力テープも入力される．しかし，その前にクリアされたレジスターへ複写される．これは図には「分枝」と記されているが，これは実際には複写過程を表している(もちろんこれは可逆操作である).

さて，逆機械 $\mathbf{M}^{-1}$ は $\mathbf{M}$ の行なったすべての仕事を元にもどして，その出力として標準のゼロと入力データを生み出す．全過程の最後に計算の答と開始時の入力の正確なコピーが残される．したがって，この素晴らしい機械はエントロピーの損失無しに計算を行ない(原理上はそうだが，実際には前に議論したようにこの系を少し駆動せねばならない)，そして可逆計算なので，われわれが仕事をしなくても計算が行なわれる．もちろん，別の計算をするためにテープをクリアするときにエネルギー損失がある．

古典的なブール論理の用語を使って考えることに慣れている人にとって，可逆計算はまったく奇妙な概念なので，君たちが可逆計算に慣れるのに役立つような問題を少し出そう．

問題 5-3 可逆コンピューターが計算を実行していて，サブルーチンを実行する必要があると仮定する．そこで，命令のまとまった組を実行させるためにある他の場所に行かせる．さて，これらの命令は，基本的な計算の要素のように，可逆でなければならず，そのため一度

このサブルーチンに入ると逆向きに走っていることに気づく可能性がある．ルーチンの最初にもどることさえ起こりうる．──この場合，置き去りにしたプログラムの本体に再び入らねばならない．さて問題は，「この同じサブルーチンが計算中に何回も使われるとすると，逆もどりが起こるときに機械はどのようにしてもどる場所を知っているのだろうか？」である．君たちに考えてもらいたい．サブルーチンを見つけに行かねばならない場所ばかりでなく，可逆なら逆にもどらねばならない場所をたどるために多くのメモリースタックを持たねばならない．サブルーチンをどのように取り扱うかという問題が可逆計算における最初の問題である．

問題 5-4　関連した問題は「if」文をどのように行なわせるかに関わっている．「if…then…」命令に従った後で，機械が逆方向に動き始めたらどうなるだろうか？　この機械はどのようにして「if」でどの方向に分岐するのかを命じたときの状態にもどれるのだろうか？もちろん，いくつかの初期状態は1つの「if」出力になるので，(「if $x=2, 3, 4$ or 6.159 let $F=d$」)，この状態を一義的に指定することはできない．このような場合を分析するうまい方法がある．各分岐で新しい変数を持ち込み，ある分岐点での各選択に対してこの変数に異なる値を割り当てればよい．君たちにこれを最後まで詳しく解いてほしい．

問題 5-5　一般的な可逆コンピューターについての簡単な疑問は，「どのくらい大きな歴史テープが必要だろうか？」である．これまで考えてきたゲートは出力の数が入力の数に等しかった．このことは可逆性にとって常に必要だろうか？　私が知る限りでは，この疑問は理論家たちによって質問されたことさえなかった．この問題に取り組みなさい．確かに出力の最小値は出力が表すことのできる可能な入力の数と何らかの関係があり，(実際に出力された答に加えて)入力を突き止めるための多くのビットが明らかに必要になるだろう．そこで問題は，まず，「理論上，ゲートを可逆にしておくために必要なビットの最小数はいくつか？」であり，そして次に，「これを実際に成し遂げられるだろうか？」である．

5-5　ビリヤードボールコンピューター

計算を実際に実行する可逆コンピューターの実例を示すために，私はここでフレトキン，トフォリたちによって発明された巧妙で簡単な機械を議論したい．この装置では，論理ゲートを通る電気信号(ビット)の動きをまねるためにビリヤードボールの平面上での運動を使う．われわれは入力を表すためにボールを機械の中に入射し，出てくるボールの分布が出力を与える．ボールはすべて平面格子の対角線上を動き，理想化された古典力学の法則に従う(すなわち，摩擦ゼロで完全

弾性衝突). 基本的な考え方を紹介するために，図 5-28 を調べよう.

この図は 2 つのボールの衝突がどのようにして 2 入力-4 出力の論理関数を表すかを図解している. このゲートに対するデータは特定の位置にボールが出現するか(1)，出現しないか(0)によって表されている. たとえば，このゲートには 2 つの入力チャネル A と B がある. もし A でボールを突くと，A の入力は 2 進数の 1 になる. もしそこにボールが無ければ，入力は 0 になる. B についても同様である. もし点 X からボールが出てくると，これは出力 X が 1 であることを意味する等々. 4 つの可能な入力状態があり，それぞれに対して初等力学を使ってこの装置から出てくるボールの配位を計算する. 4 つの可能な出力があり，そのうちの 2 つは 1 つの入力ボールが無くもう 1 つの入力ボールが直進して通り抜けてくる場合に対応し，あとの 2 つは 1 つの衝突に対応する.

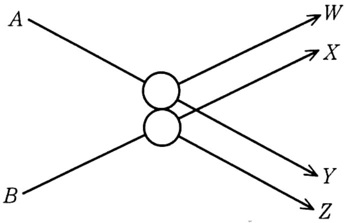


図 5-28　基本的な 2 つのボールの衝突による計算

A にはボールが無いと仮定しよう. もし B にボールがあれば，このボールは「機械」をまっすぐ通り抜けて X から出てくる. X からボールが出てくるのは，A にボールが無く，**かつ** B に 1 つのボールがある場合だけであることがわかる. 論理用語を使うと，X が 1 になるのは，B が 1 でかつ A が 0 の場合だけなので，次のように表される.

$$X = B \text{ AND } \text{ NOT } A \tag{5.34}$$

同様に，

$$Y = A \text{ AND } \text{ NOT } B \tag{5.35}$$

であることがわかる. 出力 W は少し難しい. ボールがそこから出てくるのは A と B の両方にボールがある場合だけである. 出力 Z についても同様である. したがって，W と Z の両方は同じ AND 関数を実現する.

$$W, Z = A \text{ AND } B \tag{5.36}$$

しゃれた表記法を使ってまとめてみよう(図 5-29).

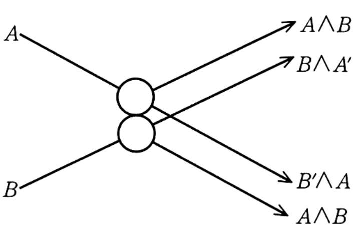


図 5-29　基本的衝突計算の論理構造

これがこのビリヤードボールコンピューターの基本的な衝突であり，論理素子がいかにうまくこの衝突から生じるかがわかる. このゲートを使って AND 以外の論理関数を作ることができる. たとえば，これを使って分枝(FANOUT)が作れる. $A=1$ として(制御線を「オン」にすることのビリヤードでの等価物) 出力を W と Z からとると，明らかにこれには B 入力の分枝の効果がある. つまり B にあるボールは W と Z の各々で 1 を作り出すが，B にボールが無ければ両出力は空白である. この装置を使って CN ゲートも作ることができる(試みなさい). しかしこの基本的な衝突ゲートは，それだけではコンピューター全体を組み立てるために十分な素子を作り出せない——2 つの線に沿って複数のボールの対が入ってくる場合には，何も変えられなくなってしまう！　ボールのコースをどのように変えればよいのだろうか？

2 つの基本的な力学的装置を導入せねばならない. 第 1 の装置は，

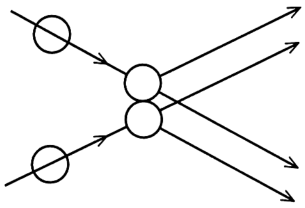


図 5-30　衝突ゲート

それが行なうことはひどくばかげたことのように思われるので，論理学者なら決して発明しないような装置である．私はこれを衝突ゲートと呼ぶ．この装置には 2 つのボールが入り 4 つのボールが出てくる(図 5-30)．これは，2 つの入ってくるボールを 2 つの静止したボールに衝突させることによって起こる「二重分枝」過程である．(このゲートのエネルギーと運動量について考えることは君たちにとって面白い練習問題である．)

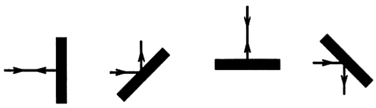


図 5-31　4 つの方向転換ゲート

第 2 の，そしてもっと重要な装置は方向転換ゲートである．これはボールを反射する鏡である．ここでは 4 方向しか向けないように制限しているが(図 5-31)，本来はこのゲートを任意の方向に向けられる．鏡を使うと多くのことができる．たとえば，鏡を使って「交差型」の装置を組み立てられる(図 5-32)．ついでだが，この装置はボールについてある重要なこと，すなわち，ボールが区別できないことを教えてくれる．われわれは個々のボールを区別して呼ばず，ボールが存在するか存在しないかだけに関心を持つ．上の交差装置は実際に入ってきたボールの向きを交換しているが，われわれは出てきたボールを区別できないので，2 つのボールはどちらも相手のそばを真っ直ぐに通過しただけのように見える．もし 1 つのボールが無ければ，もう 1 つのボールは直進するだけであることに注意してほしい．

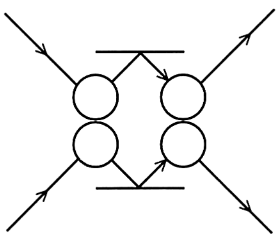


図 5-32　交差装置

これらの基本的な構造物を使ってどのような物を組み立てられるかを示すために，まずスイッチとして動作する装置を示そう(図 5-33)．これは交差型の 1 つの派生物である．B 入力があっても無くても右下の出力はつねに A と同じになることに注意すること．これは，ゲートへの入力がそのまま出力される制御線に対応する「がらくた」ビットである．もちろん，われわれはそのような出力に慣れている．

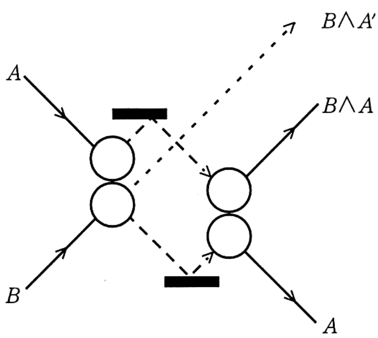


図 5-33　スイッチング装置

この章のこれまでの内容との関連で生じる疑問は，明らかに「それでは，これらの鏡とボールを使ってどのようにして可逆ゲートを作るのかを示してほしい」であり，特に，「たとえば，CN ゲートを組み立てられるだろうか？」である．これに対する答は「できる」であり，CCN ゲートもできる．しかし，すべてのゲートはフレトキンゲートだけから作られるので，フレトキンゲート，すなわち制御付交換ゲートを組み立てる方が啓発的である．フレトキンゲートは図 5-34 のようなゲートである．線 A は変化せずに通り抜ける．$A=0$ なら，B と C も変化せずに通り抜けるが，$A=1$ なら B と C は入れ替わる．私はフレトキンゲートの組み立てを練習問題にせずに説明することにする．この装置は図 5-33 に描かれたような種類の 4 つのスイッチング装置を，図 5-35 に示したようにかなり巧妙に組み合わせて作られている．明らかに，このようなコンピューターを作ることには面白いということ以外の意味はない．しかし，機械の構造をとても簡単にできることを示している．

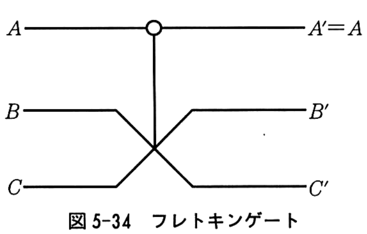


図 5-34　フレトキンゲート

さて，ボールが跳ね返る現象になじんでいる人なら誰でも，少しの誤差が急激に拡大することを知っている．テーブルの上にボールがあ

って，別のボールを上からその真ん中に落すとしよう．君たちは「えーと，ボールは鉛直に落下していくので，真上に跳ね上がり，等々」と考えるかもしれない．誰でもこれについての直観を持っているが，もし君たちが子供のときにボールで遊んだことがあれば，君たちはボールを別のボールの上でいつまでも跳ねさせられないことを知っている．これはうまくいかない！　ボールが少しでも真中からずれて斜めに跳ね返ると，次の跳ね返りはもっと外にずれてボールはもっとあさっての方向に跳び出してくる．次に落下してくるときには，ずっと真ん中からずれて下のボールにもっと斜めに当る．この次には，ボールはおそらく互いに衝突し合わないだろう．

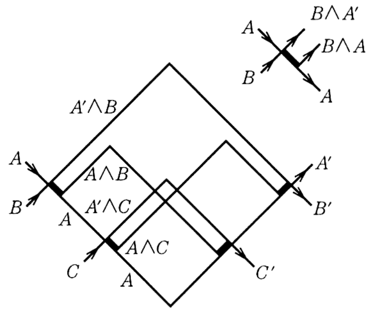


図 5-35　ビリヤードボールゲートによって実現されたフレトキンゲート

この理由を推測するのは難しくない．巨視的なレベルでは，ボールは安定でぎっしり詰まっているように見えるが，微視的なレベルでは上下左右に急速に動いている分子がごったがえしている集合体である．熱振動，統計力学的ゆらぎ等のすべては理想的なボールの単純な衝突からのずれに寄与する．事実，量子力学の極小効果でさえ割り込んでくる．不確定性原理によると，ボールの位置と運動量の両方を精密に知ることはできず，そのためにボールを完璧に真ん中に落すことはできない．理想的な 1 cm のボールが 2 つあって，1 つをもう 1 つの上の高さ 10 cm から落したとしよう．量子力学によると互いに当たらなくなるまで何回うまく跳ね返るだろうか？　われわれはこれを実際に計算でき，約 10^{17} 回跳ね返るという答になる．もちろん，現実には古典的な現象による乱れの方がずっと重要で，この量子的な極限にはとても近づけない．君たちの手でさえもブラウン運動で震えていることを忘れてはならない！

それではビリヤードボール機械という考えはまったく意味が無いことになるのだろうか？　われわれは高いところからボールを落としてはいないが，ボールを衝突させているために誤差が必然的に生じることが期待される．そこで，どのようにしてわれわれは物理的に実現可能な可逆コンピューターを持てると主張できるのだろうか．1 回の衝突あたりの誤差がわかれば，機械がばらばらになるまでの時間がわかる．10^{-3}？　5 分．10^{-6}？　よし，10 分．これは完全に絶望的に見える．この系をうまく動作させるために，ボールの動きを修正しつづける方法を見つけねばならない．ボールを窪みの中に入れることができれば，ボールを導けるかもしれない．しかし，吸収，抵抗，散逸などの損失が無ければ，ボールを窪みに入れるとボールは前後に跳ね返り，どんどん悪くなる．これらの困難に対処するようにこの窪みを設計したとしても，窪みの中の摩擦のためにエネルギーが失われるのは避けられない．われわれは機械を駆動するためにボールを引っ張らねばならないだろう．さて，少しだけこの機械を駆動すれば，駆動に必要なエネルギーは比

$$kT\frac{\text{衝突するまでの時間}}{\text{衝突したときの速さ}} \qquad (5.37)$$

の最小値になることがわかる．ビリヤードボールコンピューターの場合にこの式は詳細に解析されたことはない．

5-6　量子計算

ビリヤードボールコンピューターは主に古典力学の法則に従って作動した．しかし，これがもたらした疑問に刺激されて，人々は私に「もしわれわれのコンピューターが量子力学の法則に従って作動すれば，どのような状況になるだろうか？」と訊ねた(そして，他の人たちもこれについて考えていた*7)．極端に小さい，たとえば原子の大きさの数倍のコンピューターを作ろうとすると，このときわれわれは古典力学ではなく，量子力学の法則を使わねばならない．不確定性原理はこの計画をだめにしてしまうだろうか？　必ずしもそうではない．将来のコンピューターになるかもしれない量子コンピューターについて簡単な考察をしてこの章を終わりたい．

われわれは絶対的な極限についてもう 1 つの質問をする！　今度の質問は「コンピューターをどのくらい小さく作れるだろうか？」である．これは私が寄与をしたと思っている 1 つの領域である．飛行機とは違い，コンピューターはいくらでも小さく作れることがわかる．導線について工学的な細かいことがあるだろうし*8，われわれは出力を拡大する方法などを見つけねばならないだろうが，われわれはここではそのような実際的問題ではなく原理的問題を議論している．われわれは書き込むための何かを常に必要とするので，──実際にわれわれが必要なのは情報を伝えるビットであるが──原子*9より小さくはできない．原子または原子核はこの役目を果たす．その理由はそれらが自然界の「スピン系」だからである．すなわち，これらは測定可能な物理的属性を持ち，それらに数を付けることができ，異なる数がそれぞれ 1 つの状態を表すと考えられるからである．われわれは磁石を原子の大きさにできる．(これは何人かの化学者の仕事を奪うだろうが，それは進歩である．) 肝心なことは，大きさについて統計力学と古典力学によって課される以上の制限が，量子力学によって課されることはないということである．

私はここではあまり深入りはしない．この問題とその美しい数学には次章でもどるつもりである．今のところは，この考え方の要点だけを示す．ある理想化された量子力学系(非常に小さければ何でもよい)

*7　注目に値するのは物理学者のポール・ベニオフである(たとえば，Quantum Mechanical Models of Turing Machines that Dissipate No Energy, *Phys. Rev. Lett.* 48, pp.1581-1585[1982])．[編者]

*8　ほとんどのコンピューター理論家は導線を場所を取らない理想化された細い線として扱っていることに注意するのは興味深い．しかし，現実のコンピューター技術者は十分な導線を中に入れられないことをしばしば発見する！(第 7 章でこの議論にもどる．)[RPF]

*9　私は賢い人がもっと基本的な粒子からコンピューターを作るという可能性を予想していない！[RPF]

を考え，これが2つの状態，たとえば「上」(1つの励起状態に対応しているかもしれない)と「下」(基底状態に対応している)のうちの1つをとることができると仮定する．そうではなくて，2つの状態はこの量子力学系のスピンに関係していてもよい(スピンは粗っぽい古典的類推である)．実際には別の状態でもよいが，われわれの目的には2進数を表すために少なくとも2つの状態が必要なだけである．つまり「上」が1で「下」が0である．私はこの量子力学系を，君たちがその基本的性質を把握できるように，「原子」と呼ぶが，これはもっと複雑なものでもよいし，(2つのスピン状態を持つ)電子のような簡単なものでもよいことを覚えていてほしい．さて，このような原子を特定の仕方で一列に並べて計算装置を組み立てるというのがアイディアである．入力の2進数は，2つの状態の一方またはもう一方の状態にある原子の列で表されるが，われわれはこのような系の部分または全体から開始する．それから，われわれは系全体を量子論の法則に従ってそれ自身と相互作用させながら，時間発展させる──原子は状態を変え，1と0が動きまわり，ついにある時点でどこかで一群の原子がある状態になり，これが答を表す．

われわれはこの機械が1つの入力ビット──たとえばこの系に1個の原子を入射する──で動作を開始するように設定でき，計算が完了したとき，──たとえば1個の原子をこの系の外へ投げ出すことによって──機械自身がわれわれに完了したことを教えるように設計できる．この出力ビットが1になるまで何も信頼できないだろう．君たちはこのビットを測定し，それを0に変えて，調べるために答を凍結する．ついでに言えば，情報を出し入れすることは量子力学に特有の過程ではなく，増幅の問題である．興味深いことに，一般にはこのコンピューターが計算を完了するのに要する時間を予測することはできない．フレトキンのコンピューターのように弾道的であることがわかるが，最後に答としては波束だけが得られる．われわれは答が機械の中にあるかどうかを知ろうとする．私が設計した簡単な機械(次章参照)の場合には，いくつかの量子力学的な「振幅」(この系のある物理的な性質)が存在し，測定すればこの振幅はわれわれがどこまで計算を行なったのかを示すが，最終的には機械が計算を終えたことをわれわれに知らせるのを待たねばならない．

そこで，2050年またはそれ以前に，われわれは目で見ることさえできないコンピューターを持つかもしれない．この奇妙な新しい機械については次の章で考える．

第6章

量子力学的コンピューター

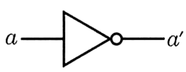
6-1 序

この章[*1]では，物理法則によるコンピューターの物理的限界に関するわれわれの分析について議論する[*2]．ベネット[1]は計算に伴う自由エネルギーの散逸について注意深く研究し，これが実質的にはゼロになることを見つけた．彼は量子力学と不確定性原理による限界の問題の研究を私に提案した．作動部分が原子でできている場合には，大きさについて明らかに限界があることを別にすると，これらの2つによる基本的限界は無いことを私は見出した．われわれはここでは理想的な機械を考えている．小さな欠陥の影響については後で考察する．この研究は原理の研究であり，われわれの目的はコンピューターとして働く系のハミルトニアンを示すことである．最も効率的な系があるかとか，どのようにそれをうまく実現できるかについては関心がない．

量子物理学の法則は時間について可逆であるため，われわれはそのような可逆法則に従う計算エンジンを考えねばならないだろう．この問題はすでにベネット[1]，そしてフレトキンとトフォリ[2]が着想し，多くの考察がなされている．しかし，君たちはこれに馴染みがないかもしれないので，ここではこれについて復習し，その際にベネット[3]の結論を非常に簡潔に復習する．その理由はわれわれが量子系の分析をするときにベネットの結論を確認するからである．

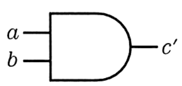
万能計算機が基本素子を相互に接続した適切に複雑なネットワークによって作られるということは計算機科学の成果である．通常の古典的な分析に従うと，この相互接続は局所的な1と0を表す1つまたは2つの標準電圧を伝える理想的な導線であると考えることができる．われわれは基本素子として2つだけ，すなわちNOTとANDをとることができる．（実際には1つの素子NAND＝NOT ANDだけで十分である．というのは，一方の入力を1にすればNANDの出力は他方の入力のNOTになるからである．）これらは図6-1に，入力導線の異なる組合せの結果の論理値とともに，記号で表されている．論理的な観点から導線について詳しく考察しなければならない．その理由

NOT



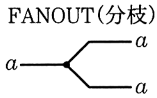
a	a'
0	1
1	0

AND

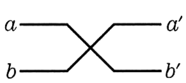


a	b	c'
0	0	0
0	1	0
1	0	0
1	1	1

FANOUT(分枝)



EXCHANGE(交換)



a	b	a'	b'
0	0	0	0
0	1	1	0
1	0	0	1
1	1	1	1

図6-1　基本素子

＊1　*Optics News*, February 1985, 11-20(©アメリカ光学会)を許可を得て収録した．読者はこの章を十分に理解するために初等量子力学を理解している必要がある．［編者］

＊2　この章の記法と文体についての取り決めのいくつかはこの本の別のところで用いられているものと異なるが，ファインマンの原論文の味わいを保つためにこれらをそのままにしておくことにした．また前の章で議論された題材がいくつか繰り返されている．［編者］

は別の系，特にわれわれが考察する量子系ではそのような導線がないかもしれないからである．われわれにはさらに2つの論理基本素子，つまり2本の導線が1本の導線に接続されるFANOUT(分枝)，そして導線が交差するEXCHANGE(交換)がある．普通のコンピューターでは基本素子のNOTとNANDはトランジスターを使って，たとえば図6-2のように作られている．

そのような基本素子から作られた理想的なコンピューターを作動するのに費やされねばならない自由エネルギーの最小値はどれくらいだろうか？　たとえば，ANDが動作する場合には，出力線c'は前の状態が何であろうとも2つの値のうちの1つに決まるので，エントロピーの変化は$k \log 2$単位になる．この事実は，温度Tでは$kT \log 2$の熱の発生を表す．長い間，この値が，計算するときに基本ステップあたりに散逸する熱の絶対最小値を表すと考えられていた．

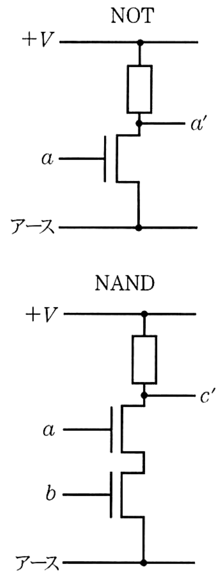


図6-2　NOTとNANDのトランジスター回路

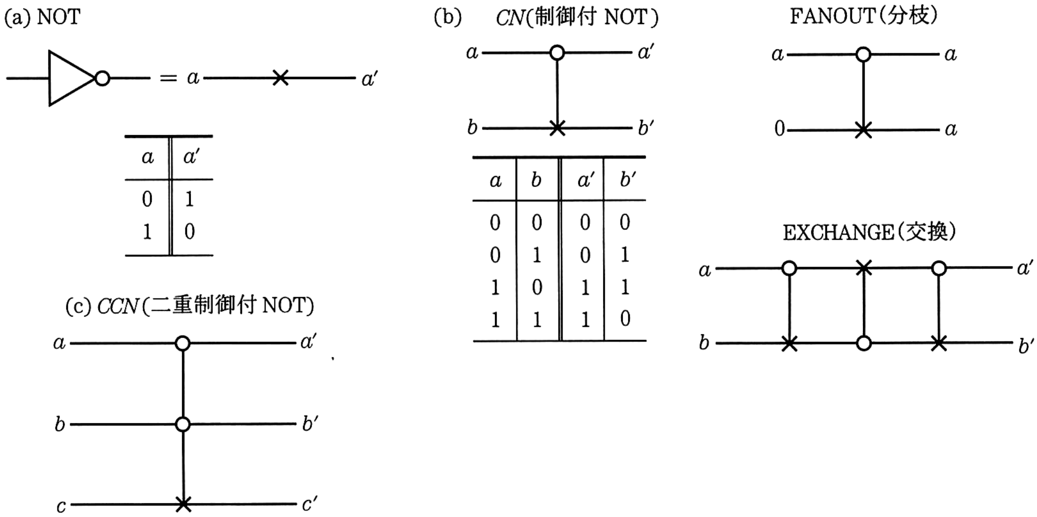
ここではこの問題は学問的なものである．実際の機械ではわれわれは熱散逸の問題にとても関心がある．しかし，実際に使われているトランジスター系では約$10^{10}kT$が散逸する！　ベネット[3]が指摘してきたように，導線の電圧を変化させるために抵抗を通して電荷をアースに落とすこと，そして導線の電圧をもう一度上げるためにもう一度抵抗を通して電荷を導線に送り込むことによってこの散逸が生じる．もしエネルギーがインダクタンスまたは他のリアクタンスのある素子に蓄えられるなら，この散逸を大いに減少させられる．しかし，現在の技術ではシリコンウェーハー上にインダクタンス素子を作るのは明らかにとても難しい．自然界でさえ，DNAの複写機では，1ビットを複写するのに約$100kT$を散逸する．現在のところは，この$kT \log 2$の値からずっとかけ離れているので，その値でさえ高すぎるとか，最小値は実際には本質的にゼロであると話すのはばかばかしく思われる．しかし後で，われわれはもっとばかばかしいこと，すなわち現在の10^{11}個の原子の代わりに1個の原子に書かれたビットについて考察する．そのようなナンセンスなことは私のような大学教授にとって非常に楽しい．私は君たちにもこの面白さがわかり，楽しんでくれることを望んでいる．

ベネットが指摘したのは，不可逆な基本素子を使わねばならないというわけではないから，この$kT \log 2$という散逸の下限は誤りであるということであった．計算は，可逆素子のみを含む可逆機械によって行なうことができる．この場合には，必要な最小の自由エネルギーは計算の論理ステップ数や複雑さには依存しない．このエネルギーは出力した答の1ビットあたりkTである．しかし，これでさえ，このコンピューターをもう一度使うためにクリアするために必要な自由エネルギーであるとも，あるいはその答について君たちがしようとしていることの一部，すなわち結果のうち他の場所へ君が伝える情報であ

るとも考えられる．この値は君たちが可逆コンピューターを使って無限小の速度で計算するときにのみ，理論上達成される極限値である．

6-2　可逆機械による計算

さて，万能機械を作るために使用できる3つの可逆的な基本素子について記述しよう(トフォリ[4])．最初は情報を明らかに失わないNOTである．もう一度NOTを作用させると逆もどりするので，NOTは可逆である．通常使われているNOTの記号は対称ではないので，その代わりに導線上の×印を使う(図6-3(a))．



a	a'
0	1
1	0

a	b	a'	b'
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

図6-3　可逆基本素子

次はわれわれが制御付NOTと呼んでいるものである(図6-3(b))．2つの入力線aとb，および出力線a'とb'がある．a'は常にaと同じであり，これは制御線である．制御が活性化される$a=1$であれば，出力b'はbのNOTになる．$a=0$であれば，bは変化せず，$b=b'$である．入力と出力の値の表も図6-3に示されている．この素子の作用は単にこれを繰り返すことによって逆にもどる．量b'はaとbについての対称関数でXOR，つまり排他的論理和と呼ばれる．これはaとbの両方ではなくどちらかだけが1のときだけ1になる．これはaとbの2を法とする(モデュロ2の)和で，aとbが異なるという信号として1を出力するので，aとbを比較するために使える．この関数XORそれ自身は可逆ではないことに注意すること．たとえば，この値が0ならば，$(a, b)=(0, 0)$によるものか，$(1, 1)$によるものかがわからないが，この曖昧さを解決するためにもう1本の線を$a=a'$にしておく．われわれは制御導線の上に○印を付けて，垂直な線で制御されている導線上の×印と結ぶことによって，この制御付NOTを表す．この素子は，$b=0$の場合にaが線b'に複写されるため，FANOUT(分枝)にもなる．この複写関数は後で重要になる．この素子はEX-CHANGE(交換)にもなる．というのは，2本の線上でこの素子を3

つ連続して使い，制御線を交互に選ぶと，2本の線上の情報の交換が行なえるためである(図6-3(b)).

これらの2つの素子のみの組合せだけではすべての論理関数を実行するのに不十分であることがわかる．3本の線を含む素子が必要である．そこでわれわれは二重制御付NOTと呼ぶことができる素子を選んだ．この素子(図6-3(c))の2本の制御線a, bへの入力はそのまま出力し，両方の線が活性化されている場合($a=1$および$b=1$)にのみ3番目の線cはNOT cに変わる．それ以外の場合は$c'=c$である．もし3番目の線への入力を0にしておくと，明らかにaとbの両方が1の場合にのみ出力が1($c'=1$)になるのでAND関数になる(後で出てくる表6-1を参照)．(a, b)の3つの組合せ，$(0, 0)$, $(0, 1)$および$(1, 0)$の場合には関数AND(a, b)はすべて同じ値0になるので，曖昧さを解消するために2つのビットが必要である．これらは出力線a, bによって保持されているので，この関数は逆にもどせる(事実，それ自身によって逆にもどせる)．AND関数はaとbの和に対する桁上げビットである．

これらの素子を組み合わせて使うことによってすべての論理回路を組み立てられることが知られており，事実，計算機科学は万能機械が製作可能なことを示している．われわれはこれを小さな例によって説明しよう．まず，図6-4に示すように，最初に二重制御付NOTを使い，つづいて制御付NOTを使うことによって加算器を作ることができる．入力線にaとbと0を入力すると，1番目の線には元のa，2番目の線には和，そして3番目の線には桁上げが出力する．

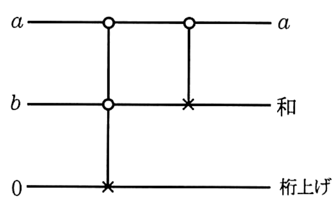


図6-4 加算器

もっと精巧な回路は全加算器(図6-5参照)であるが，これは(前の加算からの)桁上げのcを取り入れ，これに2本の線aとbを加え，そして0が入力されている追加の線dを持つ素子である．この回路には4つの基本素子を組み合わせる必要がある．全体の総和，つまりa, b, cの和と桁上げの他に，残りの2本の線には2つの情報を出力する．その1つは入力したaで，残りの1つは計算途中のある中間的な量である．

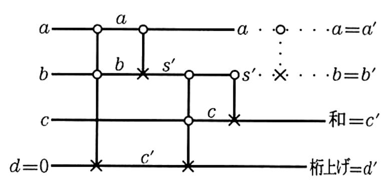


図6-5 全加算器

これは可逆系では典型的なことで，出力として望んでいるものだけでなく，ある量のごみ(garbage)も作り出す．この例の場合ばかりでなく，すべての場合に，このごみはちょうど入力と同じになるように設計できることがわかる．図6-5に点線で示されているように，最初の2本の線上にもう1つ制御付NOTを加えると，ごみはaとbになり，これらは2本の線a, bに対する入力であったことがわかるだろう．(われわれはこの回路が単純化できることを知っているが説明のためにこのようにしている.)

このようにして，いろいろな組み合せによって，nビットをnビッ

トに可逆的に変換する一般的な論理装置を作ることができる．もし君たちが実行しようとしている問題が可逆ならば，余分なごみはないかもしれないが，一般には，操作を逆にもどすのに必要な情報を蓄えるためにいくつかの余分な線がある．言い換えれば，われわれは通常の系が作れるどのような関数も作れ，さらにごみも作れる．このごみはこの処理を逆にもどすのに必要な情報を含んでいる．ごみはどのくらい出るのだろうか？　一般には，求めている出力データが k 個のビットであれば，入力と0になっている k 個のビットから開始すれば，結果として，入力と出力だけを作り出し，これ以外のごみを作り出すことはない．これは可逆である．というのはもちろん，出力と入力を知ることによってすべてを元にもどすことができるからである．この命題は常に可逆である．これに対する議論は図6-6に説明されている．

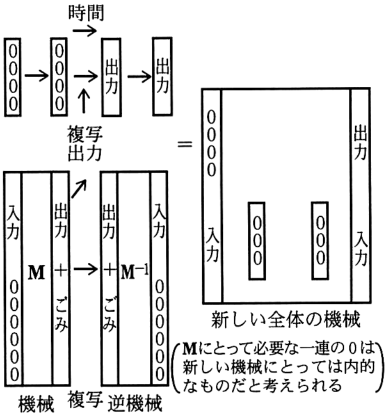


図6-6　ごみの消去

さて，入力と多数の0から開始し，望んでいる出力に加えてわれわれがごみと呼んでいるある量の余分なデータを作り出す機械 **M** を使ってわれわれは作業を始める．われわれは複写操作が制御付NOTで行なえることを知っているので，出力のために k 個のビットを持つ空白のレジスターが用意してあれば，プロセッサー **M** が処理した後で，その出力を **M** からこの新しいレジスターに複写できる(図6-6)．その後で，われわれは逆の機械，つまり **M** の逆機械を組み立てることができる．この機械は **M** の出力とごみを取り入れて，それを入力と一連の0に変える．したがって，機械を全体として見ると，出力用レジスターの k 個の0と入力から開始し，出力データが入ったレジスターと入力データを反復したものを最終結果として終了する．機械 **M** がごみを保持するために最初に必要であった一連の0は，再び0にもどされるので，新しい全体の機械(**M**, $\mathbf{M}^{-1}$ と複写)の内部にある導線だと考えることができる．そこでまとめると，われわれの目標は達成され，したがって必要なごみは入力データの反復より多くなることは決してない．

6-3 量子力学的コンピューター

次に，量子力学の法則を使って，このようなコンピューターをどのように組み立てるのかについて考えていく．万能計算機として働く大きな系と同じように振る舞う，相互作用しているいくつかの部分からなる系に対するハミルトニアンを書く．もちろん，この大きな系は量子力学に従うが，熱浴や他の物との間で，実際上は不可逆になりうる相互作用をしている．われわれはコンピューターをできるだけ小さくそしてできるだけ単純にしたいと思っている．われわれのハミルトニアンはすべての内部的な計算の動作を詳細に記述するが，もちろん，データを入力すること(初期状態を準備すること)や出力を読み取ることに関する外部との相互作用は記述しない．

このようなコンピューターはどのくらい小さくできるのだろうか？たとえば，数はどれくらい小さくできるのだろうか？ もちろん，数は1と0のビットによって表すことができる．われわれがしようとしていることは，2状態の系があると想像することであり，この系を「原子」と呼ぶ．n ビットの数は「レジスター」の状態，つまり n 個の2状態の系の組によって表される．各原子が $|1\rangle$ と $|0\rangle$ と呼ばれる2状態のうちの一方かもう一方にあるかによって，もちろん任意の数を表すことができる．そしてある瞬間に各原子がどちらの状態にあるかを決定する，すなわち測定することによって，そのようなレジスターからその数字を読み出すことができる．したがって，1ビットは2状態のうちの1つになっている単一の原子によって表され，その状態を $|1\rangle$ および $|0\rangle$ と呼ぶ．

われわれが行なわねばならないことは，二重制御付NOTの例を考えることによって理解できる．G は3つの原子 a, b, c への演算で，a, b, c の最初の状態を新しい適切な状態 a', b', c' に変換するが，その際に a', b', c' と a, b, c の関係は，a, b, c が二重制御付NOTの入力導線を表し，a', b', c' がその出力導線である場合に期待されるものであるとする．ここでは，データをある場所から別の場所へ移動させようとしているのではなく，それを変えようとしているということを理解しなければならない．実際に，導線を取り付けたコンピューターの内部で，ある導線の電圧が別の導線の電圧に切り換わるという状況とは異なり，われわれがここで作ろうとしているのはもっと簡単なもので，それは3つの原子がある特定の状態にあって，この状態を新しい値 a', b', c' へ変化させる演算が実行されるというものである．したがって，われわれが数学的形式 $|a', b', c'\rangle$ で表す状態は，単に $|a, b, c\rangle$ に演算 G を作用させたものである．量子力学では，状態を変化させる演算子は線形演算子なので，G も線形であると仮定する．したがって G は行列で表すことができ，G の行列要素 $G_{a', b', c', a, b, c}$ は，表6-1に

表6-1 G の0ではない行列要素

A	B	C	A'	B'	C'
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

おいて1である要素は1であり，これら以外の要素はすべて0である．

この表は，二重制御付NOTに対する真理値表と同じ表である．この動作が可逆であることは明らかで，この事実は$G^*G=1$を示すことによって証明できる．ここで「*」はエルミート共役を意味する．つまりGはユニタリー行列である．（このGは実行列$G^*=G$でもあるが，これは特別の場合のみである．）もっと具体的に表すために，この特定のGを$A_{ab,c'}$と書くことにする．他の基本素子を表すために異なる個数の添字を持つ同じ行列Aを使う．

簡単な例をあげると，NOTはA_aと表され，次の簡単な行列である．

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{6.1}$$

これは2×2行列であり，多くの方法，異なる記法によって表すことができるが，われわれは生成・消滅演算子の方法を使って定義する．この場合の単一の線aへの演算を考えよう．アルファベットを節約するために，次の行列を$\underline{a}$と呼ぶ．

$$\underline{a} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{6.2}$$

これは原子aの1を消滅して0に変換する．つまり$\underline{a}$は状態$|1\rangle$を状態$|0\rangle$へ変換する演算子である．しかし，原子の状態が最初に$|0\rangle$であったなら，演算子$\underline{a}$は数字0を作り出す．つまり，これは状態を変えず，その状態に作用するとき単に数値の0を作り出す．この共役演算子はもちろん，

$$\underline{a}^* = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \tag{6.3}$$

である．この演算子は，0状態に作用すると1状態へ変化させるという意味で，生成する．言い換えると，この演算子は$|0\rangle$から$|1\rangle$へ移動させる．状態$|1\rangle$に作用するときには，生成できるさらに上の状態が無いために，数字0を与える．他のすべての演算子の2×2行列は$\underline{a}$と$\underline{a}^*$によって表すことができる．たとえば，積$\underline{a}^*\underline{a}$は次の行列に等しい．

$$\underline{a}^*\underline{a} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.4}$$

これをN_aと呼ぶ．これは状態が$|1\rangle$のとき1であり，状態が$|0\rangle$のとき0なので，原子の状態が表している数を与える．同様に積，

$$\underline{a}\ \underline{a}^* = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{6.5}$$

は$1-N_a$であり，上の状態($|0\rangle$)に対して1を与え，下の状態($|1\rangle$)に対して0を与える．対角行列

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{6.6}$$

を表すのに1を使う．まとめると，$\underline{a}\,\underline{a}^*+\underline{a}^*\underline{a}=1$ が導かれる．

NOT に対する行列，すなわち NOT を作り出す演算子が $A_a=\underline{a}+\underline{a}^*$ であることは明らかである．そして，もちろん，これは可逆 $A_a^*A_a=1$ で，A_a はユニタリーである．同じようにして，制御付 NOT に対する行列 $A_{a,b}$ が求められる．制御付 NOT に対する値の表(図 6-3)を調べれば，これは次のように書けることがわかる．

$$\underline{a}^*\underline{a}(\underline{b}+\underline{b}^*)+\underline{a}\,\underline{a}^* \tag{6.7}$$

第1項中の $\underline{a}^*\underline{a}$ は線 $a=1$ という条件を選んでいて，この場合には，b に作用する NOT の $\underline{b}+\underline{b}^*$ が必要である．第2項は線 a が0だという条件を選んでいて，この場合には b には何も起こらず，b に対する演算子は単位行列であることを意味している．これは $1+\underline{a}^*\underline{a}(b+b^*-1)$ と書くこともでき，この1はすべての線を直接に通り抜けてくることを表しているが，a が1の場合には，線 b を変化しないままにしておく代わりに NOT を挿入することでこれを補正している．二重制御付 NOT に対する行列は，

$$A_{ab,c}=1+\underline{a}^*\underline{a}\,\underline{b}^*\underline{b}(\underline{c}+\underline{c}^*-1) \tag{6.8}$$

であるが，その理由は君たちにわかるだろう．

次の疑問は，一連のこのような基本素子から構成される一般の論理装置に対する行列は何かということである．例として，前に記述した(図 6-5 参照)全加算器の場合を研究しよう．さて，一般の場合には a, b, c, d と表される4本の導線があるが，すべての場合に d を0にする必要は必ずしもない(もし d を1に変えれば，d' はその NOT に変わる)．一般にこの全加算器がどのように演算するのかを述べたい．この装置は新しい数 a', b', c', d' を作り出すが，われわれはこの系の $|a', b', c', d'\rangle$ と表示される状態には a, b, c, d と呼ばれる4つの原子があり，行列 M はこれら4つの原子に作用して変化させ，この論理装置にふさわしい状態 $|a', b', c', d'\rangle$ になるようにしたと考える．つまり，$|\Psi_{\mathrm{IN}}\rangle$ を4ビットの入力状態とすると，M は4ビットの出力状態 $|\Psi_{\mathrm{OUT}}\rangle=M|\Psi_{\mathrm{IN}}\rangle$ を生成する行列である．たとえば，入力状態が状態 $|1, 0, 1, 0\rangle$ ならば，われわれが知っているように，出力状態は $|1, 0, 0, 1\rangle$ になるはずである．その理由は，最初の2つの a', b' は最初の2本の線を変化せずに通り抜けてくるために 1, 0 とならねばならず，最後の2つの c', d' は，$d=0$ なので，入力の最初の3つの a, b, c ビットの和と桁上げを表しているため 0, 1 であるはずだからである．さて，この加算器に対応する行列 M は，5つの連続した基本的演算の結果とみなせることが容易にわかるので，これらの基本的演算を表す5つの行列の積になる．

$$M=A_{a,b}A_{b,c}A_{bc,d}A_{a,b}A_{ab,d} \tag{6.9}$$

一番右側に書かれている最初の行列 $A_{ab,d}$ は二重制御付 NOT を表す．ここで a と b は制御線で，NOT は線 d 上に現れる．図 6-5 を調べる

と行列の積の残りの因子が何を表しているかはすぐにわかる．たとえば，最後の因子 $A_{a,b}$ は線 a 上の制御と線 b 上の NOT を持つ制御付 NOT があることを意味している．この積を作っている A はすべてユニタリーであるから，この行列にはユニタリー性 $M^*M=1$ がある．すなわち M は逆のある演算で，M^* は M の逆である．

そこで，われわれの一般的な問題はこういうことである．$A_1, A_2, A_3, \cdots, A_k$ が，ある論理装置のなかで n 本の線に作用する一連の必要な演算だとしよう．この目的を達成するのに必要な $2^n \times 2^n$ 行列 M は積 $A_k \cdots A_3 A_2 A_1$ であるが，ここで各 A は簡単な行列である．これらの比較的に単純な素子をどのようにして作るかを知っているなら，どのようにすればこの M を物理学的な方法で作り出すことができるのだろうか？

一般に，量子力学では，ハミルトニアン H を持つ系に対して，時刻 t での(出ていく)状態は $e^{-iHt}\Psi_{\mathrm{IN}}$ である．ここで Ψ_{IN} は入力(初期)状態である．M がこのような非可換行列の積であるとき，与えられた特定の時刻に対する $M=e^{-iHt}$ を作り出すハミルトニアンを行列自身のいくつかの簡単な性質から見つけるのは非常に難しそうに思われる．

しかし，任意の時刻に e^{-iHt} を$(1-iHt-H^2t^2/2\cdots$ のように)展開すれば，演算子 H は 1 回，2 回，3 回，などの無数の回数の演算を行ない，全状態はこれらの確率の重ね合わせによって生成されることがわかる．この事実はこれらの A の構成という問題を次のように解けることを示唆している．われわれはレジスターの n 個の原子に，まったく新しい「プログラムカウンターサイト」と呼ぶ $k+1$ 個の原子の組を加える．q_i, q_i^* ($i=0$ から k)をプログラムサイト i に対する消滅演算子，生成演算子と呼ぼう．例として，あるサイトから別のサイトへ移動する電子を考えてみよう．もし，あるサイトが電子によって占有されていれば，その状態は$|1\rangle$で，そのサイトが空席なら，その状態は$|0\rangle$である．

われわれはハミルトニアンを，次のように書く．

$$
\begin{aligned}
H &= \sum_{i=0}^{k-1} q_{i+1}^* q_i A_{i+1} + \text{複素共役} \\
&= q_1^* q_0 A_1 + q_2^* q_1 A_2 + q_3^* q_2 A_3 + \cdots \\
&\quad + q_0^* q_1 A_1^* + q_1^* q_2 A_2^* + q_2^* q_3 A_3^* + \cdots \qquad (6.10)
\end{aligned}
$$

最初に気づくことは，すべてのプログラムサイトが占有されておらず，したがって，すべてのプログラム原子が最初に状態 0 にある場合には，ハミルトニアンの中のすべての項が消滅演算子で始まっていて，ハミルトニアンが作用すると 0 になるので何も起こらないということである．

次に気づくことは，もしプログラムサイトのうちの1つだけが占有され（状態$|1\rangle$にあり），残りのサイトが占有されていなければ（状態$|0\rangle$），1つだけが占有されているという状況は常に成り立っているということである．事実，状態$|1\rangle$にあるプログラムサイトの数は保存量である．このコンピューターの作動中に，どのサイトも占有されていないか（この場合は何も起こらない），またはただ1つのサイトだけが占有されていると仮定しよう．そうすると，正常作動中に2つ以上のプログラムサイトが同時に占有されることは決してない．

サイト0が占有されていて状態$|1\rangle$にあり，残りのサイトはすべて空席で状態$|0\rangle$にあるという初期状態から始めよう．もし後のある時刻に，最後のサイトのkが$|1\rangle$状態になっているのが見つかったとすると（したがって，残りのサイトはすべて$|0\rangle$），われわれはnレジスターに望みどおりの$A_k\cdots A_2A_1$という行列Mが掛けられたと主張できる．

これがどのように実行されるかを説明しよう．レジスターは任意の初期状態Ψ_{IN}から開始し，プログラムカウンターのサイト0が占有されていると仮定しよう．この場合には，ハミルトニアンが何回も繰り返して作用するとき，ハミルトニアン全体の中で最初に作用することができる唯一の項は，初項の$q_1^*q_0A_1$である．q_0はサイト0を非占有のサイトに変え，一方，q_1^*はサイト1を占有されたサイトに変える．このように項$q_1^*q_0$は単に占有されているサイトをサイト0からサイト1へ移動させる項である．しかし，この項にはn個のレジスターの原子にのみ作用する行列A_1が掛かっているので，n個のレジスターの原子の初期状態にはA_1が掛かる．次に，ハミルトニアンが2回目の作用をすると，サイト0は今度は非占有になっていてq_0がこのサイト0に0を生成するため，初項は何も生成しない．いまここで作用できる項は，占有された点を移動させることができる項の，第2項$q_2^*q_1A_2$である．占有された点を「カーソル」と呼ぶ．このカーソルはサイト1からサイト2へ移動でき，今度は行列A_2がレジスターに作用するので，このときレジスターには作用する行列A_2A_1が掛かっている．そこで，ハミルトニアンの最初の行を眺めると，もしこの行だけしかなければ，ハミルトニアンが次々に作用していくにつれて，カーソルは0からkへと1つずつ移動し，n個のレジスターの原子に作用する行列Aが次々に得られ，全Mを望ましい順番に構成していく．

しかし，ハミルトニアンはエルミートでなければならないので，これらのすべての演算子の複素共役が現れねばならない．ある段階で，カーソルがサイト2にきて，レジスターに行列A_2A_1が作用しているとする．さて，この占有状態を新しいサイトに移動させるq_2は最初

の行から来る必要はなく，2番目の行から来るかもしれない．事実，$q_1^*q_2A_2^*$がカーソルをサイト2からサイト1へもどすかもしれない．この場合，演算子A_2^*がレジスターに作用するので，レジスターに作用する全演算子は$A_2^*A_2A_1$になる．しかし$A_2^*A_2$は1なので，この演算子はただA_1である．こうしてわれわれは，カーソルがサイト1にもどされるときの正味の結果としては，演算子A_1だけが実際にレジスターに作用したことがわかる．そこで，ハミルトニアンのさまざまな項がカーソルを前や後に移動させるのにつれて，Aは蓄積したり，減少したりする．たとえば，ある段階でカーソルがサイトjまで達していれば，A_1からA_jまでの行列がnレジスターに順々に作用している．サイトjにあるカーソルが0からjへ直接に到達したのか，あるいは先まで進んでからもどってきたのか，あるいはいろいろなパターンで前進や後退をしながら行ったのかは，最終的にサイトjに到達した限り問題ではない．したがって，もしカーソルがサイトkで見つかれば，望み通りに行列Mが初期状態に作用したというn個のレジスターの原子に対する正味の結果が得られたことになる．

それでは，このコンピューターをどのように操作できるのだろうか？　入力ビットをレジスターに入れ，カーソルにサイト0を占有させて開始する．それから，たとえば電子を散乱させることによって，サイトkが空席かサイトkにカーソルがあるかを調べる．カーソルがサイトkにあることを見つけた瞬間にカーソルを取り除いて，プログラムラインにもどれないようにする．このときわれわれはレジスターに出力データが含まれていることを知る．そこでわれわれは都合の良いときにこれを測定する．もちろん，このすべてを測定をすること，および決定することには外部の事柄が含まれており，それらはこのコンピューターの部分ではない．確かにコンピューターはデータを入力したり出力したりするために，いつかは外界と相互作用しなければならない．

数学的には，カーソルがプログラムライン上を行き来する伝搬は，演算子Aがハミルトニアンの中に無い場合の状況と正確に同じであることがわかる．言い換えれば，それは強く束縛された電子やスピン波の一次元の伝搬で馴染みがあり，非常によく知られている波である．プログラムラインを上下に移動する波があり，波束などもできる．われわれはこのコンピューターの動作を改善し，計算に使っている内蔵のプログラムラインに加えて，たとえば1列になっている多くのサイトを前後に作ることによって，このコンピューターの動作を弾道動作にすることができるだろう．これはあたかもq_iの指標iとして0未満やkより大きい値のものがあって，それらのどの項も行列Aを含まず，ただ単に1が掛かっているのと同じである．こうするとわれわれはもっと長いスピンの鎖が持てるし，カーソルを最初のサイト0に

正確に置くのではなく，ほぼ決まった値の運動量を持つ幅広い波束の初期の入射スピン波を表すように，いろいろなサイトにいろいろな振幅を持つカーソルを置いて開始することができる．このスピン波はコンピューター全体の中を弾丸のように通り抜けて，他端を出てプログラムサイトの列に付け加えた外側のしっぽに入るが，そこではスピン波が存在するかどうかを確認し，スピン波を他の場所に導き，カーソルを捕獲することは容易である．このように，この論理装置は弾道的に作動できる．

これは本質的な点であり，少なくとも計算機科学者にとっては万能計算機が作れることを示している．なぜならば，任意の論理装置を作れると万能計算機が作れることがわかっているからである．このことが素子の合成と分岐ができる万能計算機を表しているということについては，君たちにいくらかの経験がなければ完全には明白ではないのだが，私は後でこれについて少し議論する．

6-4 欠陥と不可逆的な自由エネルギーの損失

欠陥の問題のようにもっと詳しく議論をしたい問題が数多くある．この機械には欠陥を生じさせうる多くの原因があるが，最初に考えたい問題はプログラムラインに沿って結合係数が正確には等しくないという可能性である．このプログラムラインはとても長いので，実際の計算ではわずかな不規則性によって小さな散乱の確率が生じ，その波は必ずしも正確には弾丸のようには移動せずに前や後に行く．たとえば，もし系が通常の物理的な原子の基層上に作られると，これらの原子の熱振動は結合を少し変え，欠陥を発生させる．(そのような雑音は必要でもある．というのは，小さな欠陥があるとカーソルが捕らえられるかもしれない浅い落し穴が存在するからである.) そこで，カーソルの運動量がランダム化されるまでカーソルの運動量を散乱させる確率が，たとえば計算のステップあたり(すなわち，カーソル移動のステップ $i \to i+1$ あたり) p であると仮定する ($1/p$ は輸送の平均自由行程である)．p はかなり小さいと仮定する．そうすると，非常に長い計算では，波が出発してから他端を出るまでに，散乱によって何回も後や前に動かなければならないので，非常に長い時間がかかるかもしれない．そこで，われわれができることは，外部からの力でカーソルをプログラムラインに沿って引っ張ることであろう．もしカーソルが，たとえば，空席のサイトから別の空席のサイトへ移動する電子だとすると，この外力は導線に沿って電子を引っ張ろうとする電場のようなもので，導線の抵抗は欠陥や散乱確率によって生じる．このような状況では，この外力によってどのくらいの量のエネルギーが消費されるかを計算できる．

この計算は非常に簡単で，電子の平均自由行程の古典的な計算とほ

とんど同じである．カーソルが散乱されるときには，前方と後方にランダムに散乱されると仮定する．機械が動作するためには，もちろん，後方に移動するよりも高い確率で前方に移動しなければならない．したがって，散乱が生じたときのエントロピー損失はカーソルが前方に移動する確率をカーソルが後方に移動する確率で割って対数をとったものの k 倍になる．これは，(前方の確率－後方の確率)/(前方の確率＋後方の確率) の k 倍で近似される．これは散乱あたりのエントロピー損失である．もっと興味深いのは正味の計算ステップあたりのエントロピー損失であり，もちろん，これは単にこの数の p 倍になる．計算ステップあたりのエントロピー消費は次のように書き換えることができる．

$$kpv_D / v_R \tag{6.11}$$

ここで v_D はカーソルのドリフト速度であり，v_R はカーソルのランダム速度である．別の言い方をすると，これは(全ステップが常に前進方向の場合の)最短計算時間の kp 倍を実際の計算時間で割ったものである．つまり，ステップあたりの自由エネルギー損失は $kT\times p\times$「最短計算時間」/「実際の計算時間」である．この式は最初にベネットによって導出された．因子 p は，必ずしもすべてのサイトでカーソルがランダムに散乱されるのではなく，カーソルがランダムに散乱される確率が小さいという状況を表している．ステップあたりのエネルギー損失は kT ではなく，kT を2つの因子で割ったものである．1つの因子の $1/p$ は機械をどのくらい完全に組み立てられるかという程度を示し，もう1つの因子は計算にかかる時間の長さに比例する．この機械は可逆性を得るために非常にゆっくりと運転せねばならないカルノー機関にとても良く似ている．p が0であるかまたは無限に長い時間をかける理想的な機械の場合には，平均のエネルギー損失を0にできる．

ふつう，エネルギーと時間の不確定さを関係づける不確定性原理は，直接的には制限ではない．この計算機の中にあるものは計算を行なうための装置であるが，他端にカーソルが到達して出力レジスターが測定される時刻(言い換えれば，計算を完了するのに要する時間)は一定の時間ではない．これは確率の問題であり，計算にかかる時間にはかなりの不確定さがある．カーソルのエネルギーの不確定さに関連する損失はない──少なくとも計算ステップ数に依存する損失はない．もちろん，完全な機械で弾道的な計算を行ないたければ，いくらかのエネルギーを最初の波に注入せねばならないが，最終的な波がプログラムラインのしっぽから出てくるときにこのエネルギーをこの波から取り出すことができる．演算子の不確定さと測定の不可逆性とに関連する問題はすべて入力と出力の機能に関係している．コンピューター自体の量子的性質によって生じるこれ以外の制約はなく，計算のステップの数に比例する制約はない．

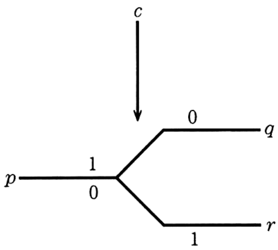
このような機械には他にも欠陥による非常に多くの問題がある．たとえば，データを保持するレジスターには混信の問題がある．これはそのレジスターの中のある原子と別の原子との相互作用，またはわれわれが必ずしも予期していなかったそのレジスターの中の原子とプログラムラインに沿って起こっていることとの直接の相互作用である．言い換えれば，ハミルトニアンの中にわれわれが書いた項の他に小さな項があるかもしれない．このコンピューターの完全な作り方をわれわれが提案するまで，この問題を分析するのは非常に難しい．少なくともこれらの問題のいくつかは，普通のコンピューターで研究されてきた誤り訂正の符号などのような技術によって通常の方法で改善できる．しかしこのコンピューターについて独特の解決法を見つけ出すまで，これらの効果の分析をどのように進めていけばよいのか私にはわからない．しかし，実際問題としてこれらは非常に重要であるように見える．このコンピューターは非常にデリケートなように思われるので，これらの欠陥はかなりの混乱を引き起こすかもしれない．

計算の1ステップにかかる時間は，ハミルトニアンの各項の相互作用の強さあるいはエネルギーに依存している．もしハミルトニアンの各項が0.1電子ボルトのオーダーだとすると，カーソルが1サイトだけ移動する時間は，弾道的に進む場合には，6×10^{-15}秒のオーダーであるように思われる．これは莫大な改善であることを表してはいない．おそらく現在のトランジスターの遅延時間の値より約4桁の改善に過ぎず，多くの光学系で達成できる非常に短い時間よりずっと短くはない．

6-5 装置の簡略化

計算を行なえる系の量子力学的ハミルトニアンを見つけるという，われわれが始めた仕事は完了した．われわれが探る必要があるのはそれだけである．しかし，装置の簡略化についてのいくつかの問題を取り扱うことはいくらか興味深いことである．われわれが書いたハミルトニアンには5つの原子間の相互作用という特殊な種類の項が含まれている．たとえば，二重制御付NOTの場合にはレジスター中の3原子とプログラムカウンター中で隣接したサイトの2原子の合計5個である．このように配列するのはかなり複雑かもしれない．問題は「もっと単純な部品を使えないだろうか？」ということである．その答は「できる」であることがわかる．各相互作用には3つの原子だけがあるようにすることができる．われわれが前に考えた基本素子ではなく，新しい基本素子を使って始めよう．NOTを使うのはよいとして，これに「スイッチ」だけを加える（プリーズ[5]参照）．

ハミルトニアンの中に，$q^*cp+r^*c^*p+$複素共役 という項があると仮定すると，図6-7のようになる．（すべての場合にアルファベッ



もし $c=1$ ならば p にあるカーソルは q に移動して $c=0$ になる
もし $c=0$ ならば p にあるカーソルは r に移動して $c=1$ になる
もし $c=1$ ならば r にあるカーソルは p に移動して $c=0$ になる
もし $c=0$ ならば q にあるカーソルは p に移動して $c=1$ になる

図 6-7　スイッチ

トの前半の $a, b, c, \cdots$ などの文字をレジスター原子に，後半の $p, q, r, \cdots$ などの文字をプログラムサイトに対して使う.）

これは，c が最初に $|1\rangle$ の状態である場合は p にあるカーソルは q へ移動し，これに対して c が $|0\rangle$ の状態である場合は p にあるカーソルは r に移動する，という意味でスイッチになっている．この操作のあいだに制御している原子 c は状態を変える．($q^*c^*cp+r^*c\,c^*p+$ 複素共役 のような制御原子が状態を変えない式を書くことも可能であるが，こうしても特に有利でも不利でもないので，簡単な方を取る.)複素共役はこれを逆にもどす．しかし，カーソルが q にあり，c が状態 $|1\rangle$ である場合には(またはカーソルが r にあり c が $|0\rangle$ であれば)，このハミルトニアンを作用すると 0 になるのでカーソルは跳ね返される．この状況が正常な操作では生じず，理想的な弾道モードで動作するように，われわれは全回路を組み立て，初期状態を選ぶ．

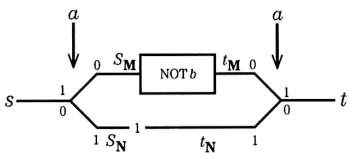


図 6-8　スイッチによって実現された制御付 NOT

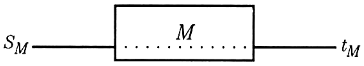
このスイッチを使うと多くのことができる．たとえば，図 6-8 のような制御付 NOT を作れる．スイッチ a は操作を制御している．カーソルが s から出発すると仮定する．$a=1$ の場合にはプログラムカーソルは上の線に沿って動き，これに対して $a=0$ の場合にはこのカーソルは下の線に沿って動き，いずれの場合にも最後はプログラムサイト t で終わる．このような図で，水平な線はプログラム原子を表す．スイッチは斜めの線で表され，箱の中には NOT b のようなレジスターに作用する行列を入れる．具体的には，制御付 NOT のこの小さな部分に対するハミルトニアンは，s から出発し t で終わると考えると，次のようになる．

$$H_c(s,t) = s_M^*as+t^*a^*t_M+t_M^*(b+b^*)s_M + s_N^*a^*s+t^*at_N+t_N^*s_N+c.c. \tag{6.12}$$

($c.c.$ は前の全項の複素共役を加えることを意味している.) ここには量子力学のあらゆる種類の複雑な特徴を作り出す可能性のある 2 つのルートがあるように見えるが，そうではない．もしこのコンピューターの全体系が，a の状態が 0 と 1 のどちらかのときに開始すれば，カーソルが t に到達するときに原子 a の状態はやはり 0 と 1 のどちらかである(ことによると原子 a に対するコンピューターの操作によって

初期状態とは異なった状態かもしれない)．したがって，2つのルートのうちの1つだけが選ばれる．この式は項$s_N^* t_N$を省略して$t_N = s_N$と置くことによって簡単化できる．この場合，上のルート(2つのカーソルサイト)が下のルート(1つのカーソルサイト)より長いことを心配する必要はない．というのはここでも干渉が無いからである．どのような場合でも一連のサイトの鎖の中に任意のサイト数の鎖を挿入したとしても，サイト間の結合が同一であれば，散乱は生じない(伝送線のインピーダンス整合に似ている)．

これらのことをさらに研究するために，部分から組み立てることを考える．部分Mとは相互作用する部品から構成された論理装置と表現することができ，図6-9にはカーソルが入るサイトs_Mと反対側の最後のサイトt_Mだけを記した．s_Mとt_Mの間にあるすべてのプログラムサイトはMの内部部品と考える．Mにはそのレジスターも入っている．s_Mとt_Mだけが外部と結合できるサイトである(図6-9)．この部分Mに対するハミルトニアンをH_Mと呼び，これを$H_M(s_M, t_M)$と書くことによってs_Mとt_Mを入力と出力のプログラムサイトの名前とみなす．したがって，H_Mは箱の中のすべての原子とその外部の開始と終了のサイトを表すハミルトニアンの一部である．



s_M＝部分が開始するプログラムサイト

t_M＝部分が終了するプログラムサイト

図6-9 1つの「部分」

考えなければならない特に重要で興味深い場合は，(正規の原子の中の)入力データがある論理装置からやってきてこれを別の装置に転送したい場合である(図6-10)．箱Mは入力レジスターの内容が0で出力レジスター(同じレジスターかもしれない)の内容も0の状態で開始すると仮定する．このときこれを次のような方法で使うことができる．たとえばs_M'から始まるプログラムラインを作る．その最初の仕事は，入力が入っている外部レジスターにあるデータを現在は0が入っているMの入力レジスターと交換することである．つまり，たとえばs_M'から始まる計算の第1段階はMの内部にあるレジスターと交換することである．これは元の入力レジスターに0を入れ，入力データを箱Mの内部の属すべき場所に入れることである．交換が終わるとカーソルはs_Mにある．(「交換」を制御付NOTから作る方法についてはすでに説明した．)それから，プログラムがs_Mからt_Mへ行くと，われわれは今度は出力が箱Mの中にあることを見出す．ここでMの出力レジスターに入っている結果を外部に用意された0の入っている新しいレジスターに書き込むと，Mの出力レジスターは消去される．このMの出力レジスターにあるデータと空白の外部レジスターとの交換をt_Mからt_M'までの間に行なう．

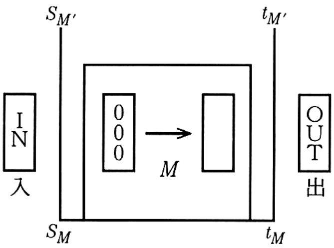


図6-10 外部からの入力と外部への出力がある部分

さて，このような装置をいろいろな方法で接続することが考えられる．たとえば，最もわかりきった方法は直列接続である．まずMを実行し，その後にNを実行したければ，図6-11のように1つの装置

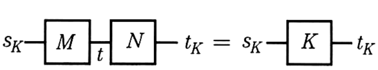


図6-11 順番に実行される処理

の終了側をもう1つの装置の開始側に接続すると新しい1つの等価な演算子 K を作ることができる．K に対するハミルトニアン H_K は，

$$H_K(s_K, t_K) = H_M(s_K, t) + H_N(t, t_K) \tag{6.13}$$

である．

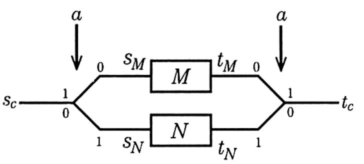


図6-12　条件付回路．もし $a=1$ ならば M を実行し，$a=0$ ならば N を実行する

$a=1$ の場合は M を実行し，$a=0$ の場合は N を実行するという一般の条件分岐は図6-12のように作ることができる．これに対しては，

$$\begin{aligned} H_{\mathrm{cond}}(s_c, t_c) = (&s_M^* a s_c + t_c^* a^* t_M + s_N^* a^* s_c \\ &+ t_c^* a t_N + c.c.) + H_M(s_M, t_M) + H_N(s_N, t_N) \end{aligned} \tag{6.14}$$

である．制御付NOTはこの回路で M＝NOT b とした特別な場合で，これに対する H は

$$H_{\mathrm{NOT}\,b}(s, t) = t^*(b + b^*)s + c.c. \tag{6.15}$$

で，N は何もしない演算 t^*s である．

もう1つの例として，(前に図6-6で記述した)ごみ消去機を考える．ここではある機械とその逆機械という2つの機械を作成せずに，われわれのスイッチを使ってデータを同じ機械に逆向きに送りもどし，1つの機械を使って実行する(図6-13)．

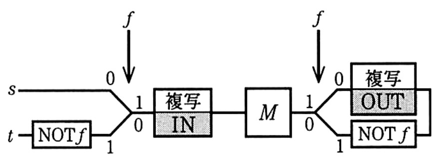


図6-13　ごみ消去機

この系には最初はいつも0にセットされている特別なフラッグ f があると仮定する．また1つの外部レジスターINには入力データがあり，出力を保持するために利用できる空白のもう1つの外部レジスターOUTがあり，すべて空白の(0が入っている)機械のレジスターがあると仮定する．われわれは開始線 s のところに来ている．最初に行なうことは(制御付NOTを使って)外部入力を M に複写することである．次に M が作用し，カーソルはこの図の上の線を行く．機械 M はその出力を外部の出力レジスターOUTに複写する．このとき M はごみを含んでいる．次に f をNOT f に変え，スイッチの下側の線にきて，ごみを消去し，入力の複写を元にもどしながら M の中を後もどりする．データを複写し再び M の計算を行なうと，レジスターのうちの1つで最初に複写したものを入れたレジスターを0にもどす．複写した後，(f は今度は変わっているので)もう一方の線に出ていく．ここでは f を元の0にもどして t に出ていく．そこで s と t の間には次のような性質を持つ新しい装置がある．この装置が動作を開始するとき，INと呼ばれるレジスターの中に入力データがある．OUTと呼ばれる外部レジスターには0が入っている．0にセットされた内部フラッグがあり，箱 M にはデータが何も入っていない．カーソルが t にやってきて，この装置の処理が終ったときには，入力レジスターにはまだ入力データが入っていて，出力レジスターには演算子 M の努力の結果の出力が入っている．しかし，M の内部はやはり空白で，

フラッグ f は 0 にリセットされている.

同じサブルーチンを何回も使える能力はコンピュータープログラムでは重要である. もちろん, 論理的な観点からは, プログラムのその小部分が使われるたびに何度も繰り返し書いてもよいが, 現実のコンピューターでは特定の演算を 1 回だけ行なう部分が作られ, その部分を何度も繰り返して使うほうがずっとよい. ここでこの可能性について示すために, まず続けて 2 回繰り返したい演算があるとだけ仮定する(図 6-14).

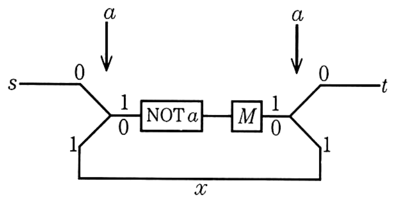


図 6-14 「M を 2 回繰り返せ」の実行

われわれはフラッグ a が状態 0 になっている s から開始し, 線に沿っていくと, 最初に起こることは a の値を変えることである. 次に演算 M を行なう. さて a を変えたために, 上の線に出ていくのではなく, 下の線に出ていく. この線はプログラムのまわりを回ってもどり, a の値をもう一度変えて, 0 にもどす. 今度は M を通り抜けて出てくると, $a=1$ なので, 上の線をたどり, 終点 t に出ていく. このハミルトニアンは,

$$
\begin{aligned}
H_{MM}(s,t) = {} & (s_N^* a^* s + s_M^*(a^*+a)s_N \\
& + x^* a^* t_M + s_N^* a x + t^* a t_M + c.c.) + H_M(s_M, t_M)
\end{aligned}
\tag{6.16}
$$

である.

このスイッチング回路を何回も使うことによって, 演算を何回も繰り返すことができる. たとえば, このアイディアをつづけて 3 回入れ子にして使うと, 図 6-15 に示したような装置によって演算を 8 回行なうことができる.

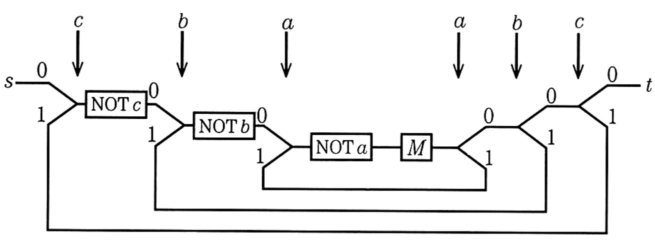


図 6-15 「M を 8 回繰り返せ」の実行

これを実行するために 3 つのフラッグ a, b, c がある. 演算が再び行なわれるときに何回これが実行され, われわれがプログラムのどこにいるのかを見失わないようにせねばならないという理由で, フラッグが必要である. そうしないと決して逆に動かすことはできないだろう. 普通のコンピューターのサブルーチンは, 何が起こったかの記録をまったく取ることなく使われ, 空白にされ, そしてまた使われることが

可能でなければならない．しかしここではわれわれはサブルーチンを使用するサイクルの中での正確な位置の記録を保持せねばならず，そのためにフラッグを使う．もしサブルーチンがある場所から呼び出され，別の場所にもどらねばならず，そして別のときに呼び出されるなら，その出所と最後の行き先とは異なっている．われわれはサブルーチンがどこから来たか，各々の場合にそれぞれどこに行くことになっているかを知り，そして見失わないようにしなければならないので，より多くのデータを保持しなければならない．可逆機械でサブルーチンを繰り返し使うことは一般の機械の場合よりも少し難しいだけである．これらの考察のすべてはフレトキン，トフォリとベネットの論文に発表されている．

メモリーの任意の場所にデータを導くことが可能なことは，このスイッチを木の枝分かれ状に次々に使えばよいことから明らかである．メモリーとは単に(プログラムにもどる前に)データを複写して入れられるレジスターがある場所にすぎない．カーソルはデータに沿ってたどらなければならない．そしてこの系が可逆であるようにデータを複写した後でカーソルを運び出すために反対の方向を向いているもう 1 つの木の枝状のスイッチの組がなければならないと私は考える．

図 6-16 には(c を一番左側のビットとする 3 ビット a, b, c による)インクリメント 2 進数カウンターを示しているが[*3]，このカウンターはカーソルが s から t へ，正味何回通過したかを記録している．これらのほんのわずかな例は，すべてのコンピューターの機能がわれわれの SWITCH(スイッチ)と NOT によって構成できることを示すのに十分であろう．このことについてもっと詳しく説明する必要はない．

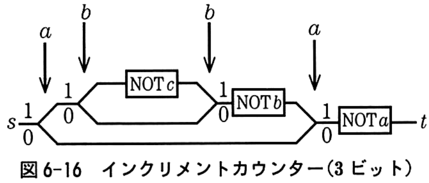


図 6-16　インクリメントカウンター(3 ビット)

6-6　結　論

この量子機械が，量子力学の微分方程式に特有な性質の多くを実際には使っていないことは，これらの例から明らかである．われわれが行なったことは，通常の逐次的なアーキテクチャのディジタル機械を可能な限り模倣しようとしたにすぎない．このことは，トランジスターのアナログで連続な振る舞いのすべてを適切に使わず，系の作用の論理解析がより簡単になるようにそれらを飽和させて，オンやオフのディジタル装置としてだけ動作させようとする通常の機械のなかのト

＊3　a, b, c の表す 2 進数は，$2^2c+2b+a$ である．〔訳者〕

ランジスターの用途に似ている．さらに，この系はまったく逐次的であり，たとえば2つのkビットの数の比較(排他的OR)でさえも，各ビットを順に比較していかねばならない．これらの可逆の量子系において，並列演算で利用可能になる速度を得るために何が行なえるかについては，ここでは検討しなかった．

理論的で学問的な理由のために，私は完全で可逆的な系を研究してきたが，もしとても小さな機械が実用的になったなら，そのような機械の操作中に不可逆でエントロピーを生成する相互作用が頻繁に生じないという理由はない．たとえば，長い計算で，カーソルが確かにある場所に到達して，そこから再び逆にもどることを許さないよう保証することは賢明であることがわかるかもしれない．(あまり頻繁に使われない項目に対しては)不可逆記憶装置を可逆論理回路または短期間の可逆記憶レジスターなどに接続することは実際的であることがわかるかもしれない．また，遠距離通信には導線や光の方がやさしくて速いので，結合したサイトの鎖に執着しなければならない理由はない．とにかく，ビットが原子の大きさになり量子的な振る舞いが支配的になるまでは，物理学の法則はコンピューターの大きさを小さくすることにとって障害にならないと思われる．

6-7 参考文献*4

[1] C.H. Bennett, "Logical Reversibility of Computation," *IBM Journal of Research and Development*, 6(1979), 525-532.

[2] E. Fredkin and T. Toffoli, "Conservative Logic," *Int. J. Theor. Phys.*, 21(1982), 219-253.

[3] C.H. Bennett, "Thermodynamics of Computation—a Review," *Int. J. Theor. Phys.* 21(1982), 905-940.

[4] T. Toffoli, "Bicontinuous Extensions of Invertible Combinatorial Functions," *Mathematical Systems Theory*, 14(1981), 13-23.

[5] L. Priese, "On a Simple Combinatorial Structure Sufficient for Sublying Non-Trivial Self Reproduction," *Journal of Cybernetics*, 6(1976), 101-137.

*4 私はT.トフォリ[4]の参考文献が役立ったことに感謝の意を表す．

第 7 章

計算の物理学的側面

編者の注意書き

本章では，この講義録で取り上げた話題の中で時とともにもっとも変化した話題を取扱っている．過去 10 年間のシリコン技術の進歩は実に驚くべきものであった．ファインマンがこの問題を考察してからいくつかの技術的目標はかなり変わってしまったが，この話題に関する 1980 年代初期の状況のファインマンによる概観を本書に含めることは，価値があると信じている．しかし，1980 年代半ばにはCMOS 技術が広く採用され，ファインマンのnMOS 技術を使ったデバイスの議論は時代遅れに見えるので，比較的複雑な nMOS の例のいくつかを編集の際に削除した．彼の CMOS デバイスについての簡潔な議論は，nMOS に比べたときのエネルギー的な有利性と省電力性に集中している．設計規則についてのファインマンの議論は，1980 年にミードとコンウェイによって書かれた VLSI についての古典的な本*1での分類では，単金属層 nMOS に限定されている．その内容を CMOS の文脈で新しいものに書き直そうと企てるよりは，ファインマンのもともとの講義を忠実に残すことにした．ただし編集の際に少し改訂した．このようにすることによって，複雑な物理的過程に貴重な物理的洞察を行なったファインマンの類のない能力がはっきり現れることを望んでいる．また，ファインマンの講義は多くの分野の専門家による講義によって補足されたことを思い出してほしい．1980 年代初期の彼のコースに参加した専門家の何人かによる最新の状況についての講義や論文を掲載する近刊の姉妹編を，ファインマンのコースのこのような部分に対応させるつもりである．さあ，読みつづけなさい．

このコースの統一テーマはコンピューターで何ができ何ができないか，そしてその理由は何かであった．われわれは機械内部の基本素子の機構から生じる制限，基礎数学によって課された限界，そして自然法則自体の結果生じる限界さえも考えてきた．この最後の章では，おそらく最も実際的な障害，つまりわれわれが機械を実際に**組み立てる**ときに使う技術から生じる制約，すなわち，われわれが使う物質と基本的な構成部品の配列法の両方から生じる制約について話す．

現在，大多数のコンピューターは**半導体**技術に基礎を置いており，この技術は機械の基本的な構成材料，つまりトランジスターやダイオードのようなデバイスを作るのに使われる．VLSI(大規模集積回路)，

*1 Introduction to VLSI Systems, by Carver A. Mead and Lyn Conway (Addison-Wesley, 1980) 邦訳は『超 LSI システム入門』菅野卓雄，榊祐之監訳(培風館，1981 年)．原書，訳書とも絶版．本章で「ミードとコンウェイの本」として引用する．［訳者］

つまりシリコンチップの作製と利用を扱うマイクロエレクトロニクスの分野，したがって計算にとって中心的重要性のある分野はそれ自体大きな対象なので，ここではただ表面を引っかく程度しかできない．エレクトロニクスの若干の知識のある読者にはこの章の理解は比較的にやさしいだろう．しかし，電気と磁気の通り一辺の知識だけの読者にとってもわれわれの説明が理解できることを望んでいる．

まず初めに，簡単なデバイスである**ダイオード**を調べる．これは気の利いたデバイスで電流を一方向のみに流す．われわれはその動作に含まれている物理現象と，それが電界効果トランジスターという工学的状況でどのように働くかを考える．

7-1　半導体デバイスの物理学

金属や他の物質の電気的性質についてのわれわれの現在の理解は，いわゆる固体の「バンド理論」に基礎を置いている．おおざっぱに言えば，この理論は，物質中の電子によって占有可能な物理状態が「バンド」と呼ばれる一連の(実際上は連続な)層に配列し，各バンドはその中で許される電子のエネルギー準位に対する特定のエネルギー範囲で特徴づけられると予言する．これらのバンドは電子と物質の格子の中に位置している親原子との複雑なからみ合いで生じる本質的に量子力学的な効果である．異なる原子状態にある電子は異なるバンドを占有する．一般の物質には，電流の伝導に関して2つの本質的に異なる型のバンド，すなわち「充満帯」または「価電子帯」と「伝導帯」がある．充満帯の状態は親原子に束縛され，物質中のある範囲に閉じ込められている電子に対応し，これらの電子は自由に放浪しない．電気伝導は電子が親原子から離れて導体中を自由に移動できるときに生じる．この型の移動できる電子は「伝導帯」の中の状態を占有するといわれる．典型的には，充満帯と伝導帯の間には**これらを分離する**エネルギーギャップがある．後でわかるように，このギャップの大きさが，その物質が導体と絶縁体のどちらに分類されるかを主に決める．図7-1に示されたエネルギーバンド構造を調べよう．

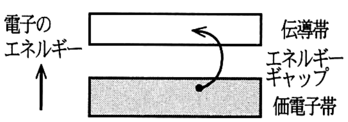


図7-1　バンド構造

この図からわかるように，エネルギーギャップで分離された価電子帯と伝導帯がある．なお，この図では上へ行くほどバンドのエネルギーが増えている．下のバンドが満員のとき，その物質は絶縁体のように振る舞う．その理由は，電場をかけたときに電子がエネルギーを得て電流を流すために利用可能なエネルギー状態がないからである．電流を流すには，多くの利用可能な空席の状態のある伝導帯に電子が存在する必要がある．そのような電子を作り出すには，価電子帯を占有している電子に十分なエネルギーを供給して，ギャップを飛び越えて伝導帯へ遷移させねばならない．この遷移のための最小エネルギーは「エネルギーギャップ」と呼ばれ，すでに述べたように，その値が物質の電気的性質を主に決定する．良い導体は通常の状態で多くの自由

電子を供給でき，エネルギーギャップはとても小さいか存在しない(充満帯と伝導帯が重なり合うことさえある)．したがって，そのような物質に電流を生じさせるのは難しくない．しかし，絶縁体には電子が遷移できないほど幅が広いエネルギーギャップ(数 eV)があり，かなり極端な条件でのみ導通する．この他に考える必要のある第3分類の物質の**半導体**がある．半導体のエネルギーギャップは比較的小さく(1 eV 程度)，半導体は導体と絶縁体のある種の複合物(ハイブリッド)である．

電子を充満帯から取り出して伝導帯に入れる主要な機構は**熱励起**である(外部からかけた電場を無視する)．これは，ランダムな熱ゆらぎによるエネルギー変動自身が，電子を遷移させるのに必要なエネルギーを供給する過程である．典型的な熱エネルギーは 25 meV のオーダーであり，これがエネルギーギャップより大きければ，遷移を起こさせるのに十分である．これは金属の場合であり，数 eV もの大きなバンドギャップを持つ絶縁体にはあてはまらない．任意の物質に対して，熱ゆらぎによって伝導電子になる確率を計算できる．物体の温度が T で，E がエネルギーギャップなら，電子が自発的に高いエネルギー帯へ跳ね上がる割合はボルツマン分布によって決まり，$\exp(-E/kT)$ に比例する．ここで k はボルツマン定数である．室温($T \approx 300$ K)では kT は1電子ボルトの約1/40，すなわち 25 meV である．この式の指数関数によって，この遷移確率は温度の増加とともに急速に上がることに注意すること．しかし，ほとんどの絶縁体では，この確率は融点のすぐ下まで無視できるほど小さい．

半導体を調べよう．セ氏0度で(そして一般に低温で)半導体シリコン(以後 Si と書く)は実際上は絶縁体である．そのエネルギーギャップは 1.1 eV のオーダーで，熱的な遷移はほとんど起こらない．しかし，価電子帯の電子にエネルギーを供給することによって電流を励起できる．そうすると，何か興味深いことが起こっているのがわかるが，それがわれわれの半導体の研究で主要なものである．電子を伝導帯へ励起させると，自由に走りまわって伝導性が生じるばかりでなく，下のバンドに**正孔**(ホール)を残す．この正孔は正の有効電荷を持ち，伝導帯の電子のように，動きまわって電流を担うことができる．つまり，もし熱で励起された粒子によって空けられた場所をすぐ近くの電子が埋めれば，その電子は元の位置に正電荷を残し，あたかも正孔が場所を移動したようになる．この正孔は「実際の」自由粒子ではなく，価電子帯にあるあたかも正電荷を持つ粒子のように振る舞う空の空間にすぎない．正孔は絶縁体にも現れるが，金属ではめったに現れない．

コンピューターで使うために理想的になるように，Si の性質を変更する特殊なトリックがある．これは**ドーピング**の処理である．ドー

ピングとは別の物質(「不純物」)の原子をSiの格子に加えることを意味する*2．よく使われる不純物はリン(P)元素である．リンは周期表の中でSiの次にある．Siの原子価は4価であるが，Pの原子価は5価である．この事実はシリコンが最外殻に4つの電子を持つのに対してリンは5つの電子を持つことを意味する．通常のシリコン結晶の格子では，これらの4つの価電子全部が格子の中の位置に原子を保つ役割をしていて，結晶の中を自由に動くことはない──価電子帯は完全に占められている．不純物P原子を入れると，各不純物原子はリンの価電子殻の5つの電子の中の4つを使って4つのシリコン原子と結合する．この結果，P原子1個あたり1個の余分な電子がSiの中を自由に動きまわり，電流を運ぶことになる(図7-2)．この結果できる物質は，過剰な**負**電荷のキャリアー(担体)があるので，「n型」半導体と呼ばれている．適度のドーピングでは，この種の物質は金属に比べて伝導度はとても低い．というのは，金属ではすべての原子あたり1つまたは2つの自由電子があるが，n型半導体は各リン原子に対してただ1つの自由電子しかないからである．

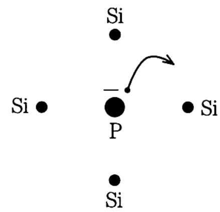


図7-2 不純物リンの電子の遊離

電子を熱的に遷移させるのに十分高い温度のときでも，n型Siにはほとんど正孔がない．その理由は，P電子は，伝導帯の準位を充たす前に，もともと価電子帯にあった正孔を優先的に充たすためである．化学反応に使われる由緒ある「質量作用の法則」は電子と正孔の密度n_eとn_hの重要な関係を与える．興味深いことに，この関係は物質中の不純物の割合によらず

$$n_e\, n_h = n_i^2 \tag{7.1}$$

である．ここでn_iはその温度での純粋な，ドープしてないSiの電子と正孔の密度である．(この関係はドープしてないSiについては自明である．というのは$n_e = n_h$にならねばならないから．) 理想的には，n_iのような物質の特性や温度がわずかであるが予期せず変わった場合でも機能する部品を設計できるようにしたい．

別の型のドーピングではいくつかのSiを周期表のグループ3の原子で置き換える．そこでわれわれはSiより最外殻電子が1個少ないホウ素(B)のような不純物原子を加える*3．そうすれば，明らかに電子より**正孔**が過剰になり，もう1つの型の半導体を見つけるだろう．電磁気学の法則の不思議によって正孔を電場の中の正電荷をもつ泡のように見なすことができ，重力場における液体中の空気の泡(実質的に「負の重量」を持っている)のように，正孔も電場の中を「逆方向」へ進む．正孔は正電荷として振る舞うため，Bをドープした Siは「p型」Siと呼ばれている．関係式(7.1)がここでも成り立つのに注意す

---

*2 ドープしてない半導体は普通は**真性**半導体と呼ばれる．ドープしているものは**外因性**と呼ばれる．[編者]

*3 ファインマンは講義の中で実際にはホウ素の代わりにアルミニウムを使った．アルミニウムは周期表の中でSiのすぐ前にあるが価電子3の不純物として使われることはきわめて稀である．[編者]

ること．

7-1-1　np 接合ダイオードと npn トランジスター

さて，半導体のどこがコンピューター部品の製造に役立つのかについて考察しよう．p型とn型のSiを互いに**接触**させるときに起こる，特に興味深い状況について調べることから話を始める．これは**ダイオード**と呼ばれるデバイスの基礎である．われわれは理想化して定性的な議論を行ない，陰気な詳細の泥沼にははまり込まないことにする．われわれは図7-3に示された状況を考える．左側には，固定された正電荷と自由に動ける負電荷から構成されたn型の物質がある．右側は逆の状況になっている．自由電子と自由正孔はそれぞれ固定された余分なP原子とB原子に対応し，P原子は1個の電子に，B原子は1個の正孔に対応するので，自由電子と自由正孔がいくつあるのかがわかる．室温では熱ゆらぎによる余分なキャリアーも存在する．

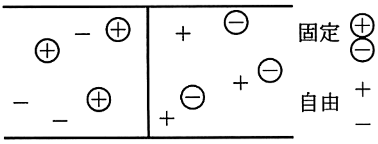


図7-3　np 接合

次に，このデバイスを回路の中に取り付けて，その両側に電圧をかける．まずその前に，そのような電場がない場合に何が起こるかを見てみよう．電荷のキャリアーはそれぞれ半分の区域で動くだけではなく，蒸気が漏れるかのように，隣の物質の中へ押し寄せる．しかし，電子がp型物質の中へ，正孔がn型物質の中へ拡散する過程は抵抗なく進むわけではない．左側の区域の固定された正電荷は逃れた電子を引きもどすような電場を作るが，この電場は正孔がn型の物質の中に浸入することによって増大する．これらの正孔も元の半分の区域に固定された負電荷や，障壁を越えて移った電子から「ぐっと引かれる」．この2つのSiの接合部で起こる4つの現象を挙げることができる．

(1)熱ゆらぎによる電子-正孔対の生成と消滅
(2)伝導(電気力によって促進されるキャリアーのドリフト)
(3)拡散(電荷密度を平坦にしようとするキャリアー)
(4)静電的過程(固定された電荷による)

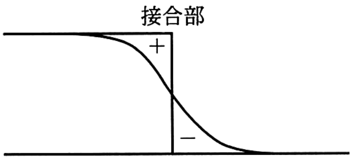


図7-4　np 接合での電荷の平衡分布

しばらく後に，この複雑な物理系は平衡状態に落ち着くが，その場合には接合部の両側にそれぞれ正と負の電荷が集中する(図7-4)．中央の領域は電荷のキャリアーがほとんどないので，**空乏層**と呼ばれる．PとBは異なる有効質量を持つので，この領域の電荷密度は完全には鏡映対称ではないが，われわれはこれらを鏡映対称だとして扱うことにする．電子，正孔，および固定電荷の符号付きの電荷密度を加えると，このデバイスの中の正味の電荷密度が求められる(図7-5)．平衡状態での物理的状況は静的ではないことを強調しなければならない．拡散による電流があるからである．しかし，平衡状態では右への電流と左への電流が相殺し，**正味の**電流は流れない．

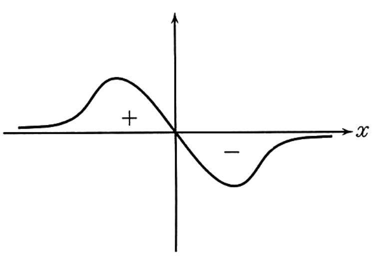


図7-5　正味の電荷密度

では，この系に電圧をかけたら何が起こるかを見てみよう．電池の

正極端子をｎ型とｐ型の物質のどちらに接続するかによって，電池の接続法には２通りある．最初にｐ型物質に電池の正極端子を接続してみよう．この場合，右側の正電位が電子をｎ型物質からｐ型物質へ引き付ける効果によって，空乏層付近の固定電荷による電流に対する反発が弱められることがわかる．電圧が増加するにつれ，ますます多くの電子が境界を越えて拡散できるようになり，もちろん，ますます多くの正孔が反対方向へ行けるようになる．率直に言えば，正しく配線をすればこの「デバイス」は猛烈に電流を流す．（この露骨な表現への重要な追加事項は，電流を維持するには電池との接点からｎ型物質へ「外部の」自由電子を引き入れることが，流れを絶えず保つために**不可欠**だということである．半導体の中の多くの「固有の」キャリアーが境界を越えると逆符号の電荷と再結合するので，補充が必要なのである．）

もし反対向きに電圧をかけると何か起こるだろうか？　さて，面白くなった――**この物質は導通しない**ことがわかる！　なぜそうなのか？　それは，ｎ型物質のすべての自由電子は接合領域から離れてうまく左へ行くことができ，ｐ型物質の自由正孔は右へ行くことができ，半導体の外に出て外部回路へ流出するからである．しかし，電圧をかけると空乏領域での電位障壁が高くなり，**ｐ型物質中の電子が空乏領域を横切れなくなる**ほど高くなる．（言うまでもなく，電池の接点から出てくるどの電子も横切れない．）同様に，正孔は右への流れを維持できず，電流は最初に一時的に流れた後は，流れなくなる．流れを維持するために利用できる自由なキャリアーは適切な場所に存在しない．われわれはこの接合に「電圧が**逆方向にバイアスをかけている**」と言い，電流が流れる状態の場合には，この接合に「順方向にバイアスがかかっている」と言う．このデバイスを**接合ダイオード**と呼ぶ．接合ダイオードは，１つの向きに電流を流すが，逆向きには流さないという基本的な性質を持つ．

接合に逆方向にバイアスがかかっているときには電流はまったく流れないのだろうか？　いや，まったく流れないというわけではない．接合部で熱的に電子-正孔対が生成するので少しは電流が流れる．対生成が起こると，電子はある向きに，正孔はその逆向きに突進する．生じる電流の大きさは明らかに温度に依存し，温度とともに指数関数的に増加するが，一般に逆向きにかけた電圧にはほとんど依存しない．われわれが希望すれば，電子-正孔対をわれわれが作ることよって，逆方向にバイアスがかかっているダイオードの熱的過程を援助できる．熱電流は普通はとても小さいので，われわれがいくつかの対を作ると，熱によるバックグラウンド以上の対を容易に検出できる．対を作る方法は半導体のエネルギーギャップの大きさに依存する．たとえば，ガリウムヒ素(GaAs)では光子(フォトン)を利用して対を作れる(この

過程は Si でもとても効果的である)．当然，この逆も正しく，電場の向きをひっくり返してダイオードが「順方向バイアス」になって電流が自由に流れるようにすれば，電子と正孔は互いに向かいあって進み，消滅して，(GaAs では)光子または(Si では)フォノンを作る．このようにして，半導体レーザーや発光ダイオード(LED)を作ることができる．

電子と正孔は n_i^2 の割合で消滅することを思い出そう．もし n_i がゼロならば，逆方向バイアスの場合には式(7.1)によって n_e も n_h もゼロなので，電流は流れない．しかし，順方向バイアスの場合には，接合部から遠く離れた p 型物質の中まで電子がどんどん充していくことによって電流が流れる．電圧の向きをもう一度ひっくり返すと，これらの電子の全部が元にもどらねばならず，このときダイオードは電流が逆向きに流れることを妨げない．つまり，ダイオードは大きなキャパシターのように振る舞う．このとき，p 型物質中の電子を接合部からますます遠く離れたところに保存するよりは，むしろ正孔に入れることによって，つまり対消滅によってこの問題は解決する．そのため，この消滅過程がないとダイオードは実際には働かないだろう．

電場によって電流が十分に流れる場合にもどろう．ダイオードの中を流れる電流 I は，かけた電圧 V の関数として計算できる．この数学はきわめて難しいというほどではないが，ここではこれに立ち入らない．この関係は非線形である．

$$I(V) = I_0[\exp(qV/kT)-1] \tag{7.2}$$

ここで V は──境界付近の固定電荷によって生じる「逆」向きの電圧を考慮に入れた──デバイス両端の有効電位差で，q はキャリアーの電荷の大きさ(単位はクーロン)，I_0 は定数である．V に対する I のグラフを図 7-6 に示す．

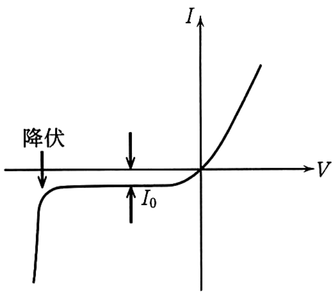


図 7-6　電圧によるダイオード電流の変化

現実の世界では，$I(V)$ は V とともに指数関数的に増加しつづけることはない．他の現象が働くようになり，接合の両端の有効電位差はかけられたものとは異なるだろう．また逆方向バイアスの場合に存在する電流の細い流れはある電圧，いわゆる**降伏**電圧で激変的に(負に)増大する．この電圧はわずか 5 ボルトから数百ボルトにまで変化でき，回路の電圧を制限する状況で現実に利用できる．

この章の次の段階はもう 1 つの半導体デバイスである有名な**トランジスター**の働きを考察することである．現在，トランジスターにはいろいろな形や大きさのものがあり，VLSI チップに広範囲に配置されているものは趣味でやっている人が使っているものとはまったく異なっている．後者の型の例として，由緒ある **npn バイポーラー接合トランジスター**について非常に簡単に述べる．これは現在使われているトランジスターデバイスの歴史的な先行物の 1 つである．このトラン

ジスターはきわめて薄いｐ型物質を２つのｎ型物質でサンドウィッチにして作ったものである(したがって，「npn」と呼ばれる)．図7-7に示されているように，3つの部分はベース，コレクター，そしてエミッターと呼ばれる．

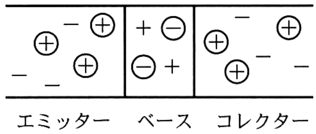


図7-7　npn バイポーラー接合トランジスターの構造

「ベース」は，トランジスターがこの物質で作られたという歴史的事実からその名前がつけられた．「エミッター」と「コレクター」という用語はともに先史時代の真空管に由来している．ベースがその両側の部分に比べて相対的に薄いことに注意しなさい．この特徴はトランジスターが正しく動作するために不可欠である．このデバイスは適切に動作させれば，**増幅器**となる．つまり，ベースへの電流の小さな変化はエミッターで増幅される．これはスイッチとしても動作でき，本書でここまで議論したトランジスター回路のすべてで使うことができる．しかし，バイポーラートランジスターは現在の VLSI チップで最も一般的に使われているトランジスターではないために，ここでは電子と正孔のレベルでこのトランジスターの動作の(複雑な)モードを議論しない．優れた議論は多くの標準的なテキストに見つけられる．その代わり，VLSI 系で最も一般的に使われている型のトランジスター，すなわち通常シリコンチップ上に作られる型のトランジスターを考察する．この MOSFET (MOS 型電界効果トランジスター)は，Metal Oxide Semiconductor Field Effect Transistor の頭文字である．

7-1-2　MOSFET

MOSFET の構造のあらましから話を始める(実際にどのようにして，そのようなデバイスをチップの上に作るのかについては後で心配することにしよう)．

シリコンチップのバルクと呼ばれる部分は，薄くドープされたシリコンの塊，つまり，その上にトランジスターなどが敷設される基板である．もしドーピングが p 型なら，いわゆる **nMOS** 技術を取り扱うことになり，もし基板が **n** 型なら **pMOS** になる．図7-8に示されているような，nMOS に焦点を合わせよう．この MOSFET には3つのポートがある．**ゲート**は伝導性のある「ポリシリコン」(金属のような物質)層であり，これはシリコンとは非伝導性の酸化物の層で分離されている．ゲートの両側には２つのｎ型の**拡散層**があり，これもゲートとは酸化物層で分離されている．(図解されているように)これらの物質はその中に拡散でドーピングされているために拡散層と呼ばれている．これらは**ソース**と**ドレイン**と呼ばれていて，npn デバイスのエミッターとコレクターに対応する．拡散層も電気を伝える．

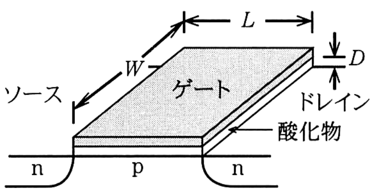


図7-8　MOSFET (MOS 型電界効果トランジスター)

トランジスターの働きはとても簡単である．np 接合ダイオードの議論から，このデバイスに外部電圧をかける前に内部で起こっていることのおおよそを理解できる．基板材質は少しｐ型で，通常はアー

スされている．基板はn型層とダイオードに似た構造を形成しているので，本質的にMOSFETは2つの背中合わせになったダイオードから作られている．前に述べたように，平衡状態ではnp接合のところに，電子や正孔が拡散でほとんど横断しない空乏層が生じる．そのため，実際上，ソースとドレインは互いに遮断されており，電圧を両側にかけても電流は流れない．しかし，ゲートに正電圧をかけると状況は異なる．この電圧の効果で酸化物の下側に電子が引きつけられる(酸化物は絶縁体なのでこれらの電子はゲートへは伝わらない)．正電位は電子の自由な移動を妨げる空乏層の障壁を下げるので，これらの電子は主にソースやドレインからくる．酸化物の下の電子はいわゆる**反転層**を形成する．**このとき**ソースとドレインの間に電圧をかけると，電流が**得られる**．つまり，反転層は本質的に2つの接合の間に電子を自由に流す通路となる．そこでスイッチというとても素晴らしいデバイスが得られる！　これを制御する電圧はゲート-ソース電圧 V_{gs} であり，この電圧が大きくなると，より多くの電荷のキャリアーがゲートの下に存在し，より大きな電流を流すことができる．しかし，$V_{gs}<0$ ならMOSFETは導通しない．この場合には，電子はゲートの下から押しのけられ，反転層は形成されない．

実際には，われわれは単純化しすぎていた．単に $V_{gs}>0$ とすることは自動的に電流を流させることにはならない．電流が流れるためには，V_{gs} がある最小電圧，つまり**しきい**電圧 V_{th} を越えることが必要である(典型的には V_{th} は $0.2V_{DD}$ のオーダーである．ここで V_{DD} は供給電圧で，たとえば5ボルトである)．このため，ゲート-ソース電圧を

$$V'_{gs}=V_{gs}-V_{th} \tag{7.3}$$

と，ずらして定義すると便利なことがわかる．これを使うと電流が流れる条件は $V'_{gs}>0$ になる．われわれは実際に，このしきい電圧が正か負のどちらかになるようにMOSFETを設計できる．負のしきい電圧は，ゲートの下にソースとドレインを接続するn型半導体の薄い伝導層ができるように，シリコンの基板をドープすることによって得られる．$V_{th}>0$ のトランジスターは「エンハンスメント型」トランジスターと呼ばれ，$V_{th}<0$ なら「デプレッション型」トランジスターと呼ばれる．(後でわかるように，デプレッション型トランジスターはnMOS VLSI内で抵抗を作るのに役立つ．)

これまでの話をもう少し定量的にしよう．あるゲート-ソース電圧およびソース-ドレイン電圧に対するドレイン-ソース電流 I_{ds} を見つけよう．後では必要ないので詳しいことに興味のない読者はここを飛ばしてよいが，ここは物理学の面白いところだ！　ゲート/酸化物/シリコンのサンドウィッチは，深さが D で誘電率が ε の物質で分離された面積 $A=WL$(ここで W はソースとドレインの幅で，L はその間

の距離)の2つの導体の板によって形成されるキャパシターと考えられる．任意の時刻でのこのキャパシターの電荷(すなわちゲートの下の電荷)を q_g と記そう．この系の静電容量 C_g は，平行板キャパシターのよく知られた公式 $C=\varepsilon A/D$ を使って計算できる．

$$C_g = \varepsilon WL/D \qquad (7.4)$$

キャパシターにかけられた電圧とこれに蓄えられた電荷との標準的な関係を使うと，次のように書ける．

$$q_g = C_g V'_{gs} \qquad (7.5)$$

まずドレイン-ソース電圧 V_{ds} が小さいと仮定しよう．電流 I_{ds} はゲートの下の電荷を電子がソースからドレインへドリフトするのにかかる時間で割ったものであることをわれわれは知っている．これは電気学の標準的結果である．このドリフトの時間はどのくらいの長さだろうか？　工学の慣行を利用すると，ドリフト速度 v_{drift} は電荷のキャリアーの「移動度」μ を使って $v_{\text{drift}}=\mu E$ と書ける，ここで E はドレインとソースの間の電場である．E は V_{ds}/L であることが容易にわかる．そこでドリフト時間は簡単にわかり，これを τ と記すと，

$$\tau = L/\mu E = L^2/\mu V_{ds} \qquad (7.6)$$

となる．この式を式(7.5)の電荷 q_g に対する式と結びつけると，電流(電荷/時間)は次のようになることがわかる．

$$I_{ds} = (\mu\varepsilon W/LD) V'_{gs} V_{ds} \qquad (7.7)$$

われわれの計算は少し単純化しすぎている．この結果はソース-ドレイン電圧 V_{ds} の値が小さい場合にのみ厳密に成り立つ．しかし，V_{ds} がかなり小さければ，トランジスターは「**その中を通る電流はかけた電圧に比例する**」という興味深い性質を持つことがわかる．言い換えれば，トランジスターは実際上は**抵抗器**として機能し($V=IR$ を思い出すこと！)，$1/V'_{gs}$ に比例する抵抗を持つ．

ドレイン-ソース電圧が増加すると，問題はもっと複雑になる．電荷のキャリアーのドリフト速度は電場 E に依存し，この電場は V_{ds} と V'_{gs} の両方によって決まる．しかし，V_{ds} が大きくなり過ぎると，トランジスターを横切って流れる電流は V_{ds} に**無関係**になり，**飽和**として知られる現象になる．このとき，電流は$(V'_{gs})^2$ に比例するようになる．この奇妙な現象は(ミードとコンウェイの本にもっと詳しく記述されている)流体モデルとの類推を使うともっとよく理解できる．もし君たちが電気より水と重力の方を好むなら，この類推が気に入るだろう！

仕切りで分離されている2つの貯水槽を思い浮かべよう．この水には慣性が無いとする──蜂蜜に少し似ていて──，そのためゆっくり流れ，あちらこちらに飛び散ることはない．初めに，図7-9に示されている状態を考えよう．ここでは可動の中央の仕切りの両側に「水」がある．

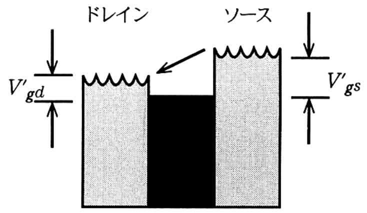


図7-9　順方向にバイアスされたMOSFETに対する流体モデル

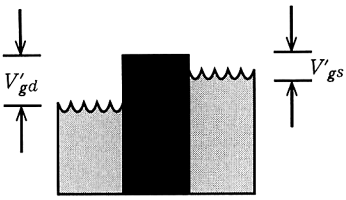


図7-10　逆方向にバイアスされた MOSFET に対する流体モデル

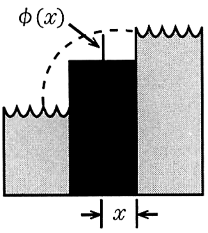


図7-11　MOSFET の飽和の場合に対する流体モデル

読者に類推を強制するために，この図には MOSFET に関連した言葉を使って注釈を付けた．この場合には，明らかに「ソース」から「ドレイン」への水の流れがあるだろう．トランジスターでは，この水の流れは電流に等しく，仕切りは電荷のキャリアーが乗り越えねばならない電位障壁の役割を演じている．この状況のトランジスターは「順方向にバイアスされている」と言われる．右側の水柱の仕切りから上の高さは V'_{gs} を，左側の水柱の仕切りから上の高さは V'_{gd} を表す．仕切りを越える流れの正確な性質はドレイン貯水槽の中の「水位」に依存する．

次に図7-10 に示されている2番目の状態を考えよう．この場合には流れはまったくない．MOSFET では，これは $V'_{gs}<0$ の「逆方向にバイアスされた」場合を表す．

3番目に飽和状態を考えよう(図7-11)．ドレイン貯水槽の水位が仕切りの高さより下になるこの場合には，ソースからの水は，**ドレインの実際の水位に無関係な**速さで，「滝」のようにドレインに流れ込む．MOSFET では，「ドレイン貯水槽」と「仕切り」が同じ高さ，すなわち $V_{ds}=V'_{gs}$ のときに，この振る舞いが起こり始める．そのとき，電流は一定になり，V_{ds} に無関係である．

飽和過程に対する洞察を得るために，この類推を物理的に実現できる．これは MOSFET を通って流れる電流の大きさのもう1つの導出法である．ゲートの下の電位 $\phi(x)$ を定義するために図7-11 を取り上げよう．Si のエネルギー準位構造によって，この電位はゲートの下の自由電子の数に比例する．さて，ゲートの下の電場はこの電位の微分 $d\phi/dx$ に比例する．したがって，電流は(電子の密度)×(電場)(この電場は電子のドリフト速度を制御している)に依存せねばならないため，次のようになる．

$$I = K\phi(x)d\phi/dx = (K/2)d\phi^2/dx \tag{7.8}$$

ここで，K は定数である．さて，考えてみると，電流 I は一定で，x に無関係でなければならないことがわかるだろう．これは ϕ^2 が x について1次で，したがって $\phi\propto\sqrt{x}$ とならねばならないことを意味する．そのため，次の一般的な式が得られる．

$$I = (K/2L)[\phi^2(0)-\phi^2(L)] \tag{7.9}$$

しかし，飽和時には $\phi(0)=V'_{gs}$ および $\phi(L)=0$ なので，以前に述べた I の V'_{gs} についての2次の依存性を回復する．飽和時でなければどうなるだろうか？　式(7.7)をもう一度導けるだろうか？　実際にはノーである．しかし，もっと良いものが得られる．不飽和の場合，$\phi(L)=V'_{gd}$ はゲート-ドレイン電圧で，次の式が得られる．

$$I \propto (1/2)(V'_{gs}+V'_{gd})V_{ds} \tag{7.10}$$

ここで，$V_{ds}=V'_{gs}-V'_{gd}$ である．なぜこの式が改善になっているのだろうか？　その理由は，この式は前の式(7.7)の異常性を取り除いたものになっているからである．前の式には，電流 I にソースとドレ

インの間での奇妙な非対称性があった．直観的には，式(7.7)のゲート-ソース電圧をゲート-ソース電圧とゲート-ドレイン電圧の平均値で置き換えたい──そうすると，まさに式(7.10)になる．参考のために，この節で考察したいろいろな電圧に対する電流の変化を図示する(図7-12)．

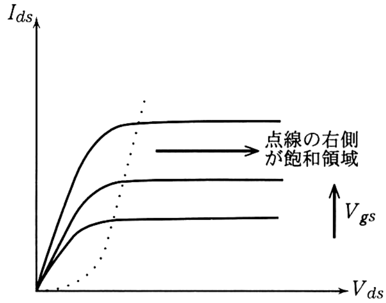
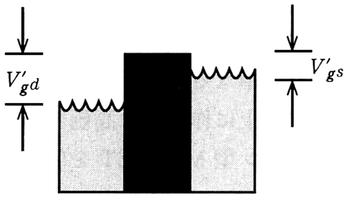


図7-12 MOSFETの電流-電圧の変化

MOSFETについてのもう1つの質問は「電流が流れない状態では，このデバイスはどのくらいよい絶縁体か？」である．君たちは接合ダイオードでは逆方向にバイアスされたときにも熱によって誘起された小さな電流が流れることを思い出すだろう．トランジスターでのこの効果について簡潔に議論しよう．逆方向バイアスの場合に対する「水」の図解，図7-10を思い出しなさい．



電子がソースからドレインへ飛び越える確率は電子が電位障壁V'_{gs}を乗り越えるのに十分なエネルギーを持つ確率，すなわちexp$(-qV'_{gs}/kT)$に比例せねばならない．したがって順方向電流もこの因子に比例する．また逆方向電流もある．同様にこれはexp$(-q[V'_{gs}+V_{ds}]/kT)$に比例することがわかる．したがって，全電流は，$V'_{gs}<0$のとき，次のように与えられる．

$$I = (\text{定数})\cdot\exp(-qV'_{gs}/kT)[1-\exp(-qV_{ds}/kT)] \tag{7.11}$$

そのため，順方向にバイアスされた場合から逆方向にバイアスされた場合へ切り換えても，電流は瞬間的に消えるわけではない(デバイスの温度が室温よりおそらく高いことに注意)．しかし，$kT\sim 1/40$ eVなら，約5ボルトで動いている電気回路の電流はすぐに消える．図7-12を逆方向電流も含めるよう修正できる．

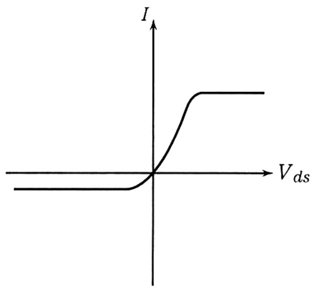


図7-13 ソース-ドレイン電圧の関数としてのトランジスター電流

MOSFETの性質について学んだことを検討してまとめよう．まず，このデバイスを議論するのに，これを含んだ回路を描くたびに図7-8の物理的な図を描き直すよりも，図記号を使う方が便利である．VLSI技術で出会うさまざまな異なったトランジスターについて多くのこのような記号がある．図7-14はMOSFETについてのかなり共通な規約の図記号で，われわれはこれを採用する．この図記号はn型かp型のトランジスターを表している．(どちらであるかを示すためにこの図を修正する方法については後で示す．これは両方の型を使用するCMOS技術では必要である．) 両者に対する規則は，これらがスイッチのように働くということである──ゲート電圧V_gが十分なだけ正になれば，トランジスターは導通する．n型のMOSFETに対しては，次の規則がある．すなわち，AとBのうち電位が低い方

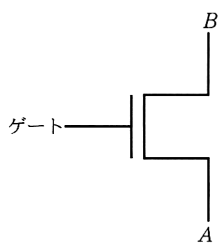


図7-14 慣習的なMOSFETの記号

をソース，他方をドレインと呼ぶことにすると，ゲート電圧がソース電圧より高い，あるしきい電圧 V_{th} よりもっと高い場合は，このデバイスは導通する．つまりスイッチは「閉じ」て電流が流れる．p型のデバイスについては，A と B のうちで電位が高い方がソースで，ゲート電圧がソース電圧より**低い**，あるしきい値よりもっと**低い**場合はトランジスターは導通する．以前にMOSFETの2つの動作の型，つまり**エンハンスメント**型と**デプレッション**型を定義した．前者はしきい電圧 V_{th} がn型では正の場合で，p型では負の場合である．デプレッション型では話が逆になる．そして，デプレッション型MOSFETの良い特徴は，$V_g = V_s$（ソース電圧）なら，デバイスは常に導通することである．その結果，ソースとゲートを直接に接続して自動的に同じ電圧になるようにすれば，トランジスターはもはやスイッチとしてではなく**抵抗器**として働くことがわかる（図7-15）．なぜこのようにするのだろうか？　それは単なる経済性と設計の問題のためである．標準的な抵抗器をチップの上に作るのは高価であり，そして大いに場所をとるが，デプレッション型MOSFETを使うとこのどちらの障害もなくなる*4．（ついでながら，このトリックはエンハンスメント型デバイスではうまくいかないことに注意．）

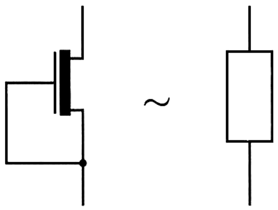


図7-15　抵抗として働くデプレッション型MOSFET

問題7-1　さて問題を出そう．間もなく私は答を実際に示すが，君たちはこれらの事柄の本質的な部分の理論を練習するために数学を使って解きたいだろう．この問題は静電容量と関係がある．ダイオードでは，電流ばかりでなく，静電容量も電圧の非線形関数である．MOSFETについても同様である．さて，静電容量のほとんどは酸化物の中にあるが，全体は実際には極度に非線形になっていることがわかる．とても興味深い！　私が君たちに与えようとしている問題はこの非線形性の性質を例で示すように作られている．

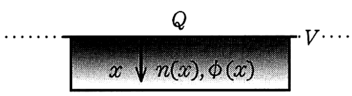


図7-16　ゲート-半導体電極の接触の図式的モデル

MOSFETのゲートの電極の接触のモデルを図7-16に示したように作る．薄くドープした大きなp型物質があると仮定しよう．理論上は，この物質は無限の深さ（図の中で垂直にとっている）になっている．この表面（水平な面）にとても大きな金属板を置く．極板上に正電圧 V を与えると，負電荷のキャリアーがその下に引き寄せられる．極板からの静電気力が無視できる，物質の深いところでは，負のキャリアーの数はドープされた不純物イオンの数にちょうど等しくなる．しかし，極板の電圧のために極板の近くのキャリアーの密度は大きく，極板から離れると減少していき，一定な値になる．物質の中への深さを x で表すと，位置に依存するキャリアーの密度 $n(x)$ を定義できる．また結果として生じる物質内の電位 $\phi(x)$ も定義できる．はじめの正

*4　（後に出てくる）CMOS技術の出現に伴い，今ではnMOSデプレッション型トランジスターが使われるのは稀である．これらはpチャネルエンハンスメント型トランジスターで置き換えられている．［編者］

の不純物濃度を n_0 で表そう．さて，答えてもらいたい問題は「電極にはどのくらいの電荷があるだろうか？ 言い換えると，この物理的な設定の静電容量はいくらか？」である．

君たちの解答を助けるために，いくつかのヒントを与えよう．まず，$\phi(0)$（すなわち，電極での電位）を極板の電圧 V とし，理想化された値 $\phi(\infty)$ を 0 とせねばならない．熱力学的な考察から得られる $n(x)$ と $\phi(x)$ の関係を君たちにサービスして教えよう．

$$n(x) = n_0 \exp[q\phi(x)/kT] \tag{7.12}$$

である．ここで，q は負のキャリアーの電荷で，T はいつものように温度である．n_0 は定数である．もう 1 つの本質的な関係は物質の中に深く入っていくときの $\phi(x)$ の変化率（言い換えれば，半導体内部の電場）と極板の電荷密度 Q の間の関係である．ここで電荷**密度**を定義しているのが重要であることに注意すること——**無限に広い**極板の場合に**全**電荷を議論することは無意味である．ガウスの定理から $x=0$ で

$$\partial\phi/\partial x = Q/\varepsilon \tag{7.13}$$

という結果が得られる．ここで ε はドープされた物質の**誘電率**で，極板から離れるのにつれてどれくらい速く電場が減少するかを決める．標準的なポアッソン方程式 $\partial^2\phi/\partial x^2 = -\rho(x)$ を使い，境界条件を使って積分し，$kT/q=1$ および $n_0 q\varepsilon = 1$ という単位を使うと，式(7.14)の形の結果が得られる．

$$Q = V[2(e^V - V - 1)/V^2]^{1/2} \tag{7.14}$$

ポアッソン方程式の右辺の $\rho(x)$ は物質内部の電荷密度（これは $n(x)$ と N で表される）である．平方根の中の V^2 を平方根の外の V と相殺すべきだと君は考えるかもしれないが，これは Q に対する正しい符号を得るために必要なのである．さて，静電容量を定義する標準的な公式 $Q=CV$ と比べると，この系の静電容量は極板電圧 V について極端に非線形の関係を示すことがわかる．私の知る限りでは，この性質は VLSI ではあまり利用されていない——しかし最近「ホットクロッキング」（これについては後で議論する）で応用されている．

ここまでは，シリコン基板上の孤立した MOSFET を考察してきた．VLSI の核心部への旅の次の段階では，論理回路を作るためにこれらのトランジスターをチップ上でどのように組み立てるのかを考察する．いまわれわれは現実の機械のところに来ている！

7-1-3 MOSFET 論理ゲートと回路素子

論理回路を組み立てるには，まず論理ゲートを組み立てることができる必要があり，われわれはすでに第 2 章で一般の「トランジスター」を使って論理ゲートをどのように作るかを見た．MOSFET の場合も同じようにして作る．図 7-17 のように，抵抗を通してトランジスターに電圧 V_{DD} を供給すると何が起こるかを考えよう．

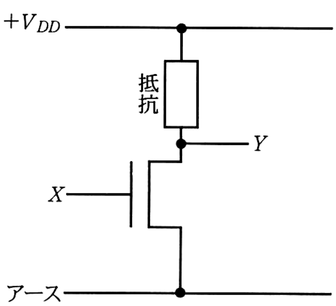


図 7-17 反転器回路

このトランジスターがエンハンスメント型で動作している nMOS 型のものだとする.（VLSI 設計には多くの型があり，それらのすべてを考えることはできない——特定の 1 つに焦点を合わせることには意味がある.）もし端子（ゲート）X の電圧がゼロに近ければ，トランジスターは絶縁体で，Y での出力電圧は供給電圧 V_{DD} に近い．つまり，われわれはこの状況を出力 Y が論理の 1 だと解釈する．しかし，X の電圧が V_{DD} に近ければトランジスターは導通する．これが抵抗よりずっと良く導通すれば Y の電圧はゼロに近くなり，この状態を論理の 0 だとみなす．通例，これらの極値の間で動作させることは，おそらく一時的な場合以外はない．したがって，この単一 MOSFET デバイスは，入力信号をひっくり返すために，第 2 章で見た NOT ゲート（反転器）として働く.

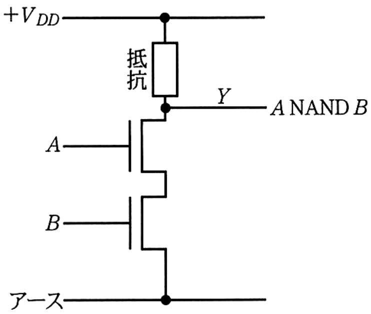


図 7-18　MOSFET によって実現された NAND ゲート

残りの基準的なゲートについては，第 2 章の先例にならうことができる．たとえば NAND（NOT　AND）ゲートは図 7-18 のように組み立てられる．この系で出力 Y が 0 になるためには，入力 A と B がともに論理の 1 でなければならない．AND ゲートを手に入れるには，この出力に反転器を付ければよい．NOR ゲートをどのようにして手に入れるか思い出すためには，第 2 章を調べなさい！

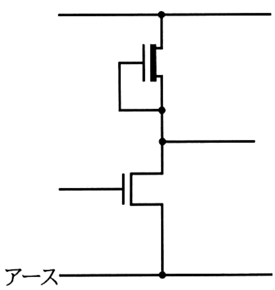


図 7-19　MOSFET 抵抗を使った nMOS 反転器

論理ゲートのほかに，MOSFET を使って他の役立つ素子をチップ上に作ることができる．抵抗器を考えよう．前に述べたように，標準の型の抵抗器をシリコンチップ上に付けると高価で面積がかさむので，デプレッション型トランジスターを抵抗器として使うことは一般の慣例になっている．そこで，nMOS 技術では，われわれの反転器の MOSFET 構造は実際には図 7-19 に示されているようになる.

ところで，厳密な論理的考察からは明白でない MOSFET のもう 1 つの本質的な性質がある．それは**増幅器**としての振る舞いである．図 7-20 のように 2 つの反転器をつづけて置いたら何が起こるかを考えなさい.

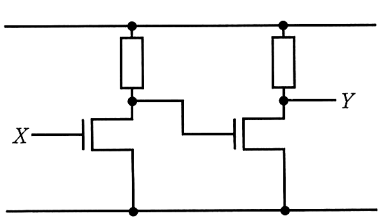


図 7-20 「フォロア」回路

論理的な観点からは，これは単に恒等を作り出したという，とてもつまらない動作である．つまり，何の計算もしない．しかし，機械装置の観点からは，注意深くなければならない．トランジスターはエネルギーを散逸するので，このようにデバイスを鎖状につないだ場合（縦続接続），各デバイスで電力が減り，出力は次第に減少して最後には無くなると単純に考える人がいるかもしれない．そうなると実に悲惨だろう！　しかし，このようなことは明らかに起こらない．というのは，2 番目のトランジスターへの入力電流は少し減るかもしれないが，それはこのトランジスターの動作モード（すなわち，導通または非導通）を変えるほどではなく，その出力 Y は従来どおり供給電圧まで引き上げられる（またはアース電圧まで引き下げられるかのどちらか適切な方）．言い換えると，出力は常に確定した論理的な判定を表

し，鎖に沿った小さな電力のゆらぎに比較的に鈍感である．この回路はきわめて有効ないわゆる「フォロア(従者)」であり，線の後方の電力，またはインピーダンスを引き上げる(これを二重増幅器と言ってもよい)．ある意味では，しっぽを制御するだけで犬全体を制御できる．言うまでもなく，この増幅性は何千または何百万というトランジスターを含む回路の首尾よい動作にとって決定的で，この回路では回路を通り抜ける信号を回復させることが絶えず必要になる．増幅があることはすべての計算技術にとって不可欠である．

VLSI では，計算の他分野と同じように，われわれはしばしば同期の問題に関心をいだく．この点に関連して，反転器の動作の速さについて質問するのは興味深い．この質問は「鎖状に接続した反転器の最左端の入力にスイッチを入れれば，右側の出力に何が起こるか？」である．確かに瞬間的にはスイッチしないだろう．というのは，各トランジスターの出力は次のトランジスターの入力に送られてそのゲートを充電せねばならず，これには時間がかかるからである．有効静電容量 C_g を持つ各ゲートの電圧がある値 V だけ変化しなければならない．この過程にどのくらいの時間がかかるかがわかり，この速度を上げる方法を考えつけば，われわれはもっと良い機械を手に入れられるだろう．この過程は図 7-21 に描かれている回路を調べることによって解明できる．なお，ゲートの静電容量を表すためにこの図にわざとキャパシターを挿入した．

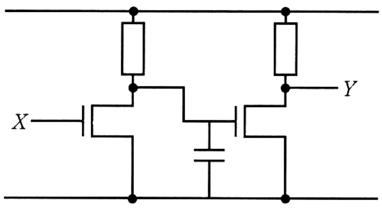


図 7-21 フォロア回路に類似した電気回路

判定(すなわち，トランジスターがスイッチするのに十分なゲート電圧)に必要な蓄積電荷を Q としよう．そうすると，$Q=C_gV$ である．この電荷をどれほど速く届け，または運び去ることができるだろうか？ まず，状態 $X=1$ は最初のトランジスターの出力が正確にアース電圧になっていることに対応しない．というのはトランジスターはある最小抵抗(これを $R_{\min}$ と呼ぶ)を持つので，その両端の間でわずかな電圧降下が生じるからである．ところで図 7-22 に示されている標準的な RC 回路との類似を仮定し，放電時間を積 $R_{\min}C_g$ と決めることはエレクトロニクスの標準的な結果である．標準的な回路理論によると，キャパシターの電荷 Q の時刻 t での値 $Q(t)$ は，次式に従う．

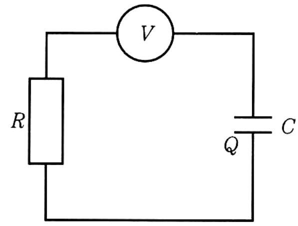


図 7-22 等価な RC 回路

$$Q(t) \propto \exp(-t/R_{\min}C_g) \equiv \exp(-t/\tau) \qquad (7.15)$$

ここで，$\tau=R_{\min}C_g$ である．

もし反転器をもっと速く動作させることだけに興味を持てば，R と C の両方を減少させることによってこれを達成できる．明らかに，これは回路をもっと小さく作ればできる．しかし，それには限界がある．というのは，活動していない状態でさえ，ソースとドレインからの電子は MOSFET のシリコン基板の中にわずかな距離だけ浸み出しているからである．デバイスを縮小していくと，これらのキャリアーは反対の極へどんどんドリフトしていき，ゲートの下の領域を実際にショートさせてしまい，ゲート電圧を操作しなくても電流が流れてしまう．

このようなことが起これば，製図板にもどらねばならない．つまりトランジスターはもはやこれまでのように動作しないので，再設計が必要なのである．これは自然がわれわれの技術に制約をどのように課すかについてのよい例である！

それではわれわれがもっと小さな機械を組み立てたければ何をすればよいのだろうか？　その答は，規則が変わるときには，すでに述べたように，再設計することである．たとえば，非圧縮性空気と低速飛行機の航空工学の場合を考えよう．詳しい解析によると，プロペラ機には「音速の壁」があり，音速を越える速度では飛ばないという結論が出た．音速より速い飛行機を手に入れるには，ずっと前の振り出しにもどる必要があった．現在，Si コンピューターの大きさの基本的な限界は見つかっていない――音速の壁の類似物はない．この問題は他の人たちとは別の考え方をすることが実を結ぶかもしれないという1つの実例である――君たちはまごまごと進んでいるうちに何か新しいものを見つけるかもしれない！　現在，最新技術のデバイスでは RC の値は約10ピコ秒である*5．この値を大きく下げる技術の開発に君が成功したときには，おそらく別の人が別の技術を使ってもっと良いもの作っているだろう．実際にこのようなことが超伝導計算デバイスで起こった．研究者たちが超伝導計算デバイスを研究していたときに，従来の VLSI 技術で進歩があるたびにこのデバイスの有利性は絶えず消滅していった．このようなことはまったくありふれた出来事である．

ここまでは，計算に使われるいろいろな半導体デバイスの構造を概観してきたが，この分野での現実的な限界についてはほとんど述べてこなかった．これから現実的な限界のいくつかの例を話すが，コンピューターの熱発生と電力損失という重要な話題の議論から始める．

7-2　コンピューターのエネルギー使用と熱損失

第5章で，われわれは典型的なトランジスターが1回のスイッチあたり約 $10^8 kT$ の熱を散逸することを指摘した．これは現象論的な量である――もしこれを1桁または2桁下げることができれば，すべてのファン(扇)を取り除けるだけでもこの機械をかなり簡単にできる！　いままで議論してきた nMOS 技術について特に悩む問題は，MOSFET の動作の安定した状態――たとえば $X=1$ $(Y=0)$ でトランジスターがこの値を**保持し**変えない場合――でも**電流は絶えず流れる**ということである．そのため，たとえこのトランジスターが何もしなくても，電力を捨てている！　明らかに，もっと経済的な動作が望める技術は

*5　ファインマンが本コースを開講したときには，RC の値は実際には4ナノ秒であった．このタイミングの400倍の改善は VLSI 技術が異常な速度で進んだことの例証になっている．［編者］

何であっても探求する価値がある．そしてわれわれがこの節で考察しようとしている**相補型 MOS** (CMOS, Complementary Metal Oxide Semiconductor)技術はまさにそのような技術である．

7-2-1　CMOS 反転器

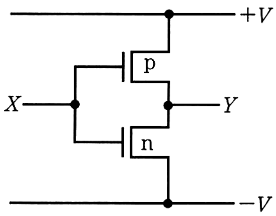


図 7-23　CMOS 反転器

CMOS では，回路に n 型と p 型の MOSFET を混ぜて使う．n 型と p 型の MOSFET を組み合わせて標準的な CMOS 反転器を作る方法を図 7-23 に示す．nMOS 反転器と同じように，ある電圧 V に対して，論理の 1 は $+V$ の近くだとするが，論理の 0 はアース電圧ではなく $-V$ に選ぶ．各 MOSFET のドーピングの型(n または p)を示すために，記号に隣接して適切な文字を書くという規約に従った．nMOS デプレッション型トランジスターが通常の p チャネルトランジスターで実際上置き換えられていることに注意すること．この回路は組み立てる価値があるだろうか？　その答は，次のような理由で，イエスである．入力 X が正だとしよう．このとき，n 型 MOSFET はソース電圧より高いゲート電圧を持ち導通する．他方，p 型デバイスは逆方向にバイアスされているので導通**しない**．出力 Y は $-V$ まで下げられる．次に X を $-V$ にスイッチする．図 7-23 からわかるように，今度は上のトランジスターが導通し，下は導通しない．電圧 Y は供給電圧まで上がる．これは反転器の標準的な動作にすぎないので，ここまでは新しいものは何もない．しかし，この回路は目新しい特徴を持っている．つまり，具体的に言うと，遷移が起こった後は**この回路には電流は流れない！**　絶縁性の n 型 MOSFET によって $-V$ への経路は遮断される．(入力を元にもどしたとき何が起こるかは読者に考えてもらいたい．)

これは注目すべき性質である．CMOS 反転器では，状態を**保持するため**にエネルギーは必要でなく，状態を**変化させる**ためだけにエネルギーが必要なのである*6．この CMOS 反転器は，論理ゲートのエネルギー論を調べるための有効で簡単な「実験室」としても役立つ．どれだけのエネルギーが論理処理に必要かという問題は第 5 章での要約で考察したが，実際的な問題を論じることは明らかに重要である．あたりまえだが，われわれはデバイスが機能するのに最小のエネルギーしか使わないことを望んでいる．そしてこの目的のために，判定に必要なエネルギー量，スイッチングにかかる時間，構成要素の信頼性，その大きさなどを考慮しなければならない．鎖の部分としての CMOS 反転器の電気的な振る舞いをもっと詳しく考えることから始めよう．そうすると CMOS デバイスの増幅性についても調べられる．話を進めるためにミードとコンウェイによる単純化された(そしてあまり正確でない)モデルを使おう．

---

*6　厳密に言えば，逆方向バイアスのトランジスターを流れる小さな電流があるが，われわれの考察ではこれを概ね無視する．[RPF]

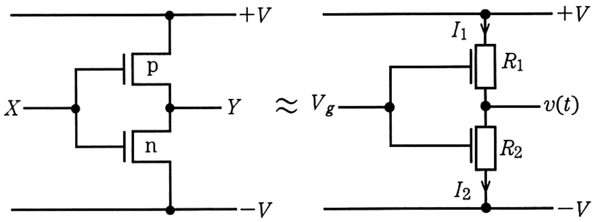


図7-24　CMOS反転器の単純化されたモデル

このモデルでは，2つのトランジスターを単に制御された抵抗器として扱う．したがって，次のような等価関係がある(図7-24)．

このCMOSデバイスは，直線状につづいているものの中の1つとして見るべきものである．つまりXへの入力信号は前のゲートから入ってくる．そして，出力Yは次のゲートへの入力と考えるべきもので，次のゲートはアースに対して有効静電容量C（たとえば)を持っている(この静電容量を，厳密には正しくないが，一定とする)．最終的にわれわれはXでのゲート電圧$V_g(t)$を変化させるとき，すなわちスイッチを実行するときの出力電圧の振る舞いを調べたい．最初に入力ゲートに対する電圧を一定に保つという簡単な場合を考えよう．この電圧は電流の流れを促す．出力の最終平衡電圧はどうなるだろうか？　トランジスターを通る電流をI_1およびI_2と表し，この2つの差(すなわち，Yに接続された次の構成部分へ電荷を移動させる電流)を$I=I_1-I_2$と定義する．Yの電圧は時間の関数で，たとえば$v(t)$である．Yに蓄積している電荷を$Q(t)$と表す．標準的な回路理論から

$$dQ/dt = I_1-I_2 = Cdv(t)/dt \tag{7.16}$$

であることをわれわれは知っている．そして式(7.11)から，小さなドレイン-ソース電圧に対して電流I_nが次式によって与えられることがわかる．

$$I_n \approx V_{ds\,n}/R_n \tag{7.17}$$

ここで，$V_{ds\,n}$の解釈は明らかであり，有効抵抗R_nは次式によって与えられる．

$$R_1 = R_0 \exp(qV_g/kT), \qquad R_2 = R_0 \exp(-qV_g/kT) \tag{7.18}$$

V_gでのしきい電圧のサイン(′)をこっそり除いてしまったことに注意すること──われわれはこのデバイスをいくぶん理想的なものと考えている．

さて，I_1とI_2に対する基本的な式

$$I_1 = (V-v)/R_1, \qquad I_2 = (v+V)/R_2 \tag{7.19}$$

を式(7.16)，式(7.18)と結びつけると，$C, v(t)$とRを関係づける微分方程式

$$Cdv/dt = -(2V/R_0)\sinh(V_g/V_T)-(2v/R_0)\cosh(V_g/V_T) \tag{7.20}$$

が導かれる．そこで，ゲートの電圧を固定すれば，出力での平衡電圧

の値，すなわちすべてが落ち着いたときの値はどうなるだろうか？すべてのものがピシャピシャと動くのを止めたときには，$dv/dt=0$ なので，平衡値 v_e が次のように与えられることがすぐにわかる．

$$v_e = -V\tanh(V_g/V_T) \qquad (7.21)$$

ここで，v_e は定数である．典型的な場合に V_g/V_T は大きな正の数あるいは大きな負の数なので，平衡電圧は漸近的に $+V$，または $-V$ に近づく．

この結果を使って，鎖状につながったCMOS反転器の増幅性を解析できる．ゲート入力を少し変え，たとえば $V_g \to V_g + \delta V_g$ とすると，これに対する応答として出力が変化する．この変化量を $\delta v_e = A V_g$ と記す．これに対する応答として v_e を入力されたゲートの出力，たとえば v' が，$\delta v' = A\delta v_e = A^2 V_g$ だけ変化する．鎖に沿ってこのようなことがつづく．このCMOSデバイスが作動するなら，この因子 $|A|$ の大きさは明らかに**1より大きく**ならねばならない．そうでなければ，鎖の左側での入力信号のどのような変化も左端から右端まで伝搬せず，先細りになって消えてしまう．増幅因子 A は v_e 対 V_g のグラフの原点 $V_g=0$ での傾きである(図7-25)．原点での傾きは(君たちが導けるように) $-V/V_T$ である．したがって，鎖状につながった系が作動するためには，約1/40 eV以上の直流電圧を供給するだけでよい．もちろん，実際には供給電圧はずっと高いので(たとえば，5または6ボルト)，増幅は著しい．小さな入力信号の変化が出力で何倍にも増大されるので出力電圧は入力に極度に敏感な関数である．

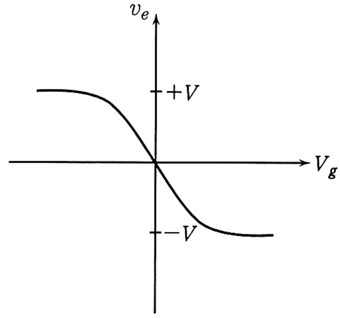


図7-25 CMOS反転器でのゲート電圧対出力電圧

問題7-2 ここにあるいくつかの問題は君たちにはやさしいものではない．いままで，われわれはCMOS回路の平衡状態の振る舞いを考えた．ここで君たちにやってもらいたいのは時間的な振る舞いの解析であり，式(7.20)を解いて，入力のスイッチを入れたとき出力が切り替わるまでにどれくらいの時間がかかるかを求めることである．V_g が時間の任意関数の場合の一般解は明らかに難しすぎるので，計算では入力電圧は**無限に速く**切り替わると仮定しなさい．次に，反転器内でのエネルギーの散逸を考えなさい．平衡状態では回路を電流は流れないと考えることは有効な定性的理想化であるが，これは実際の状況ではないと前に述べた(事実，前の計算ではそうではないとした！)．逆方向にバイアスされたトランジスターは非常に高い抵抗を持っているというだけである．この結果，絶えず小さな電力損失が生じるが，この損失は抵抗 R での電圧降下による損失電力に対する標準的な電気の公式を使うと V^2/R であることがわかる．ここで R は「非伝導の」抵抗である(代わりに I^2R を使うこともできる．ここで I はリーク電流である)．スイッチング過程でも電力損失があり，これはゲート上の電荷を(いま導通している)抵抗を通して放出するときに起こる．スイッチング間のエネルギー損失は $2Cv_e^2$ であることがわかるはずである．ゲートの有効静電容量の時定数 τ はいくらか？

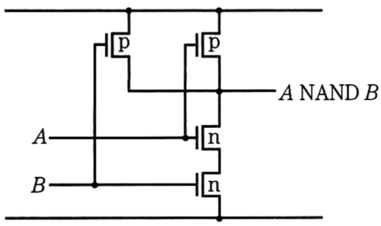


図 7-26　CMOS で実現された NAND ゲート

われわれは主として VLSI のエネルギー論について教えてくれる CMOS 技術に興味があるが，この講義を完全なものにするために脱線して，CMOS 反転器を使ってどのように一般的な論理ゲートを作るのかを例で示そう．NAND ゲートの作り方を考えなさい——一般的な論理ゲートのうちの1つを組み立てられたなら，すべての論理ゲートを組み立てられることを思い出しなさい．NAND ゲートは図 7-26 に示した配列で作れる．これがどのように動作するのか見てみよう．NAND ゲートでは，両方の入力が1の場合は出力は0になり，その他のすべての入力の場合には1になることを思い出しなさい．明らかにこれはここで起こることである．すなわち，図 7-26 の**下の2つのトランジスターが導通するときだけ**出力電圧は $-V$ へ下がる．すなわち，論理の0になる．このことは2つの入力が正のときにのみ起こる．もし2つの入力のどちらかが負であれば，対応するトランジスターは導通しなくなり，その出力は $+V$ にとどまる．

CMOS デバイスでのエネルギー散逸の問題にもどろう．実際にはスイッチング1回あたりのエネルギー散逸は $10^8 kT$ のオーダーである．これは非常に大きいので，ここには工学の世界をあっと言わせる機会がある．**なぜそれほど大きいのかという理由がない**からである．明らかに，電圧はこのデバイスで使われている技術によって決まる大きさでなければならないが，これは基本的な制約ではなく，散逸するエネルギーを減らすことは可能なはずである(第5章の解析では $kT \log 2$ が理論的に達成できたことを思い出すこと)．この領域で何ができるかを議論しよう．

スイッチング過程で実際に起こっていることを考えよう．スイッチングする前には，入力の静電容量に電圧がかかっていて，そこにはある量のエネルギーが蓄えられている．スイッチングすると，電圧は逆になるが，静電容量のエネルギーは同じエネルギーである．ということは，われわれは回路からすべてのジュースを取り出して下水に捨てて，すぐ後で電源から再充電して，あるエネルギー状態から同じエネルギー状態にするという愚かなことをしてしまったわけである．これは高速道路を猛スピードで運転し，ブレーキをぐっと踏んで——大きなブレーキ音を出して！——停止し，それから逆方向に後向きになってまたスピードを出して車を走らせることに似ている．時速 60 マイルで出発し，また同じ速さにもどるが，その過程でものすごく大きなエネルギーを散逸する．ここで，理論上は自動車のエネルギーを(たとえば)はずみ車(フライホイール)に与えてエネルギーを蓄えることが可能なはずである．その結果，止まってから，新しいエネルギー源からではなく，はずみ車から動力を取り出すことによって再出発させることができる．エネルギーを廃棄する必要はない．VLSI にはこのはずみ車の類似物があるだろうか？

1つの提案は慣性の電気的な類似物のインダクタンスにエネルギーを蓄えることである．エネルギーを投げ捨てないように回路を組み立て，「箱の中に」保存して，後で取り出せるようにすることは可能だろうか？ 調べてみよう．インダクタンスの概念を説明するために，水を使ったもう1つの有用な類推を使う．君たちの中で電気に興味をもつ者は水と電気の類推に慣れているし，電気よりも力学の方が気楽に感じる者にも水を例にとればやさしいことがわかるだろう！ 図7-27に示したような2つのパイプが接続された大きな水の容器があるとしよう．各々のパイプは本質的に無限に大きな貯水槽(示されていない)につながっていて，上のパイプを通ってそこから水が流入でき，下のパイプを通ってそこへ水が流出できる——この流れは各パイプのバルブで調節される．ここでの類推ではパイプとバルブがトランジスターを表し，貯水槽の水は放出されるのを待っている電源の電荷である．上の貯水槽は電圧 $+V$ に，下の貯水槽は電圧 $-V$ に対応していて，タンクの水の高さは電荷が放出される電圧と解釈できる．この類推が意味をなすようにしておくために，バルブの1つが開いている(導通している)場合は，もう1つのバルブは閉じている(絶縁している)ようになっている．反転器でのスイッチング過程をモデル化するために，この系のバルブを開閉して何が起こるかを見てみよう．

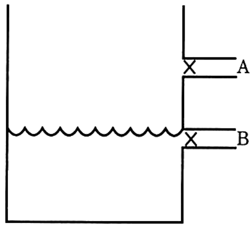
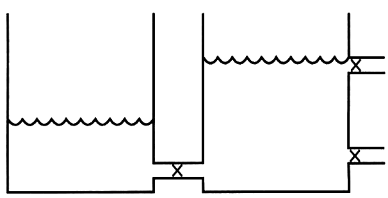


図7-27 CMOSスイッチに対する水の類似物

初期条件は図7-27に示されており，上のバルブが閉じ下のバルブが開いている．水はある平衡状態の水位になっている．ここでこの系を切り替えるために，下のバルブを閉じて(正ゲート電圧に対応する)上のバルブを開ける．上の貯水槽の水は奔流して入り，——ジャブ，ジャブ！——新しい平衡の深さになる．その過程で騒音，摩擦，乱流などが起こり，エネルギーが散逸し，最終的に，新しい平衡点に落ち着く．今度は初めの状態にもどりたい．そこで，下のバルブを開け上を閉じて，切り替える．タンクの水位は下がり始め，タンクの水はいろいろな方法でエネルギーを散逸しながら，元の高さまで下がる．初めの状態にもどったが，その間にとても多くのエネルギーを使い果たしてしまった！

われわれは装置を改造して，切り替えるたびにこれほど大きなエネルギーを失わないようにしたい．その1つの方法は次のようなものである．別のタンクを最初のタンクの隣に置き，バルブの付いているパイプでこの2つのタンクを結ぶ(図7-28)．上のバルブを開けたので，元のタンクの水位が図7-28のようになったとしよう．今度は上のバルブを閉じて連結したタンクへのバルブを開けると，水はジャブジャブとパイプを通って新しいタンクへ入っていく．このタンクの水位が最高になったときにバルブを閉じる．連結バルブを開けたままにしておけば，水は2つのタンクの間をパチャパチャと往復し，2つのタンクの水の高さが同じである状態に落ち着く．この場合に圧力は等しく


図7-28 CMOSスイッチに対するエネルギー節約の類推

なるが，安定するまでにかかる有限な時間は水には慣性があるという事実の結果として生じる．このバルブが最初に開いたとき，新しいタンクの水位は(系をパチャパチャさせつづけた場合の)平衡値より高いところに到達する．同じように，元のタンクの水位は平衡値より低くなるだろう．最高点に到達直後に連結バルブを閉じると，水のほとんどと，したがってそのポテンシャルエネルギーを新しいタンクの中に捕えることができる．もちろん，その全部ではない──摩擦などによる損失があり，新しいタンクに少し水を足さねばならないだろう．さて，水のエネルギーを取りもどしたければ，右側のタンクが干潮になっているときにこのタンクへの連結バルブを開くだけでよい．

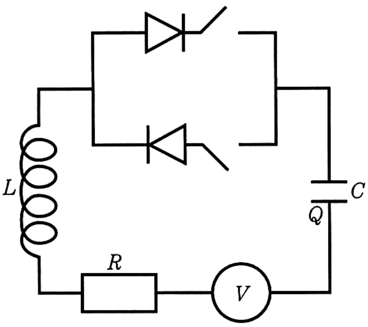


図 7-29　誘導回路

このアイディアをシリコンで実現するには，これに対する電気的な類似物が必要であり，これは慣性の類似物が必要なことを意味する．すでに述べたように，電気での類似物はインダクタンスである．上のアイディアを実現する１つの方法は図 7-29 のような回路を考えると理解できる．

この回路はキャパシター，インダクタンス，抵抗そしてダイオードを使った２つの「チェックバルブ」を含んでいる．スイッチの１つを閉じると，ダイオードは電流を一方向にのみ流すので，水のモデルでの２つのパイプを通る一方向への水の流れを真似している．君たちはこの回路の振る舞いを定義する基本方程式をよく知っているはずだ．

$$Ld^2Q/dt^2 + RdQ/dt + (Q/C) = V \qquad (7.22)$$

である．ここで，V は回路の両端の電圧である．CMOS を使ってこの種のアイディアを実現できるかどうかを調べることは君たちに任せたい．あいにく，シリコン技術を使ってかなりの容量のインダクタンスを作ることはきわめて難しいことがわかっている．長い導線やコイルが必要だが場所がない！　そこでこれはエネルギー損失を下げるための実用的な方法ではないことがわかる．しかし，このことは基本的なアイディアを捨てねばならないことを意味しない．試行できる非常に気の利いた方法は，スイッチごとに１つずつというように多くの小さなコイルを使うのではなく，チップから離れたところにあるただ１つのインダクタンスを使うことである．

7-2-2　ホット-クロッキング

エネルギー散逸を減らすためのまったく異なった，非常に賢い方法がある．それは**ホット-クロッキング**と呼ばれている技術である．この方法では，**電源電圧を変える**ことによってエネルギーを節約しようとする．このような方法がどのように，そしてどうしてうまくいくのだろうか？　水との類推にもどろう．前にわれわれは，タンクの水位が低いときに上のバルブを開くと，水が上からどっと入ってきて滝のように落ちるのでエネルギーを失うことを知った．不適切だったのは**水位が違うときにスイッチを開いた**ことである．そうすればエネルギ

ーが失われるのは不可避である．しかし理論上は，エネルギーをほとんど浪費せずにタンクに水を注入する別の方法がある．たとえば，1つのスイッチのあるパイプが付いているタンクがあり，このパイプの他端は貯水槽だとしよう．図7-30に示されているように，スイッチを開いて水を注ぐときにパイプが常に水の高さになるようにパイプを上に移動させ，徐々にタンクを満たせば，エネルギーは散逸しないだろう．もちろん，エネルギーを散逸する滝となることを**完全に**避けるために無限にゆっくりと実行せねばならない(この型の議論は第5章でしばしば使った)．しかし，とてもゆっくり実行し，パイプとタンクの水面に高さの差があるときにはスイッチを開かなければ，エネルギー損失を減らすことができる．電気学には，「**スイッチの両端に電圧がかかっているときには決してスイッチを開閉するな**」という類似の原理がある．このことは，まさにわれわれが行なってきたことである！

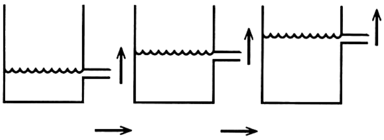


図7-30 エネルギーを散逸させずにタンクに注水する

ホット-クロッキングの基本原理はこうである．図7-31の修正された反転器回路を考えよう．この図で，上と下の電圧 V_{TOP} ($=V$，たとえば)と V_{BOTTOM} ($=-V_{\mathrm{TOP}}$)は一定と考えては**いけない**．つまりこれらの電圧は可変であり，また実際に変わるので，気をつけなさい！われわれは2つの主要な電圧の状態を定義する．そのうちの**静止**状態は上の電圧が負で下が正の状態に対応する．**ホット**状態はこの反対で，上の電圧が正で下が負の状態に対応する(これらの割り当ては任意であり，状態は逆にしてもよい)．このデバイスの動作原理はこうである．われわれは静止状態から始めるので，図7-31の上の電圧は負で，そして X を正($=+V$)にすると仮定しよう．その結果，p-MOSFETは開き，n-MOSFETは閉じ，電流は流れない(n型のデバイスには電圧がかからない)．事実，X が負でもダイオードの整流特性によって電流は流れないだろう．そこで，静止状態では入力を行きあたりばったりに切り替えることができる．つまり，回路は入力電圧にまったく鈍感である．そこでわれわれは明らかに Y の初期状態を自由に選べるので，これを正にとろう．

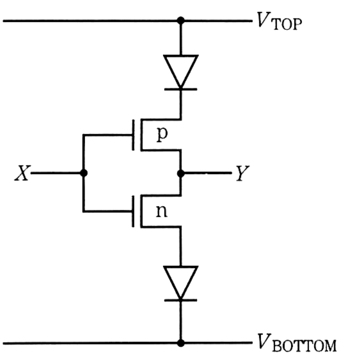


図7-31 「ホット-クロッキング」回路の例

次に，電圧をホット状態にする──われわれは徐々に電圧をひっくり返す．そのとき下のダイオードにかかる正電圧は徐々に増大し，ダイオードは導通する．そこで出力 Y は下の電圧(これはいまは負である)まで下がる．ダイオードの抵抗は小さいのでこのときのエネルギー散逸は小さい．下の電圧が底値になり物事が落ち着いたとき，われわれはまた静止状態にもどす．このときダイオードは電流が流れるのを妨げるため，出力 Y は前の値にもどりたくてももどれない．静止状態では X を変えられる．すなわち，この安定な状態では入力を好きなように切り替えられる．このサイクルの最初の部分を，Y が変わるときは，ゆっくりと行なう必要がある．しかし2番目の段階の静

止状態への復帰は迅速に遂行できる.

ところで，Y の出力を他のゲートに入力せねばならない．明らかに，出力が変化しているときにはこれは使えないので，次のゲートの電圧サイクルは，(2位相のクロックのように)別の電源も使って，いくぶんかずらして起こるようにしなければならない．フリップ-フロップではよくあるように，2番目の信号を単に最初の信号の逆にして，1つの電源だけを使うことは可能である．しかし，静止状態へ復帰するときに Y が少し変わるので，これは危険で少し混乱させる．2つの別の電源を使って，堅実に設計する方が安全である．2つの電源の電圧変化を図示しよう(図7-32).

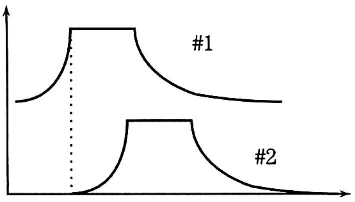


図7-32　供給電圧

各パルスの立ち上がりはたれ下がりよりゆるやかで，2段階のスイッチング時間の差を反映していることに注意すること．また，これらの電源はチップの全体または複数のチップについて共通であることを指摘しよう．そうでなければ，供給電圧を変えるのに必要なエネルギー量はわれわれが得る節約分と実際上相殺してしまうことになるだろう．われわれは流出するエネルギーを電源部分に蓄えているのである.

ダイオードの配列にもどってスイッチングの間に失われるエネルギーを計算しよう．供給電圧が変化する「立ち上がり時間」を t と仮定しよう．この変化の間に移動せねばならない電荷は $Q=CV$ であり，したがって流れる電流は(平均で) $Q/t=CV/t$ になる．ダイオードを閉じたときの抵抗値 R は，トランジスターの抵抗と同様に，小さいと仮定すれば，エネルギー損失の割合，すなわち電力損失は $P=I^2R=Q^2R/t^2$ である．したがって，スイッチングでの全エネルギー損失は,

$$\Delta E = Pt = C^2V^2R/t = (CV^2)(CR)/t = (CV^2)\tau/t \tag{7.23}$$

である．ここで，τ は元の単純なCMOS反転器回路の時定数である．また CV^2 はその回路のスイッチングでのエネルギー損失であったことを思い出そう．したがって，エネルギー損失とこの損失が生じる時間の積は古い回路と新しい回路の両方で同じであることがわかる．この事実は各スイッチングまたは単純な論理演算に対して，次の形の一般的な関係があることを示唆する.

$$(\text{エネルギー損失})(\text{損失時間}) = \text{定数} \tag{7.24}$$

この式は「ゆっくり進めばそれだけ失うエネルギーが少なくなる」という第5章の結果と確かに調和しているように見える．実際の回路では，クロックはトランジスターよりずっとゆっくりしているので(たとえば，約50分の1)，クロックすることは計算する上で大量のエネルギーの節約を可能にする．あいにく，現在はかなりスピード狂時代なので，この電力節約の利点は十分に利用されていない．たしかに，機械は遅くなり，そして余分な構成部品のために大きくなるが，この

欠点は，より安く運転でき，機械を冷却するために必要なポンプやファンなどがかなり不要になるという利点によって償われるだろう！

さて，図 7-31 の例で私はダイオードを使ったが，いろいろなタイプの素子を使いたくなければ，もっと現実的な構成は図 7-33 に示されたものであることを指摘せねばならない．この構成ではダイオードはトランジスターで置き換えられている．

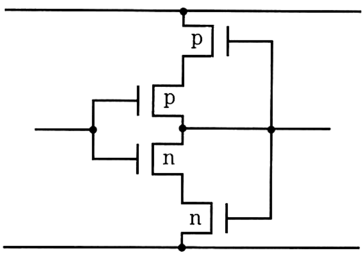


図 7-33 「ダイオードを使わない」ホット-クロッキング回路

われわれはエネルギー散逸を減らすために，いわゆる「ホット-クロッキング」法のうちの 1 つだけを考察した．（大部分はカルテク*7で開発された）この技術はクロックを伝える線が電力を送ることを可能にしているが，もともとは時間をエネルギーに交換する意図はなかった．ホット-クロッキングは最近発展した技術で，未解決の問題が多いので，君たちがこの分野で実際に何かを行ない，貢献する機会があることを指摘してこの節を終えたい．私が描いた回路は私自身の回路で，これまで設計され作られてきた他の回路とは違うが，私は自分の回路が他の回路に比べて利点があるかどうかはよくわからない．しかし君たちはあらゆる種類のアイディアを詳しく検討できる．たとえば，供給電圧が交流，すなわち，正弦波だったらどうなるだろうか？両方とも交流だが位相の違う 2 つの電源を使うことができるだろうか？ 論理素子にかける電圧をなぜ交流にしないのか？ おそらくわれわれは電源と同位相のもの（論理の 1）と位相のずれたもの（論理の 0）の 2 つの状態を定義できるだろう．多くの機会があり，君たちがもっと深く探求しつづければ，何か興味があることを発見するかもしれない．

7-2-3 一般的考察と興味深い関係

前節の主要な発見の 1 つは，スイッチングに必要なエネルギーとスイッチングにかかる時間の積が——少なくとも抵抗系については——定数であるということである．これは一般的な関係かもしれない．この定数を「散逸作用」（私がたったいま作った新しい言葉）と呼ぼう．さて，反転器の典型的な時定数 τ は 0.3 ナノ秒のオーダーであり，これはかなり小さい．このように小さくなければならないのだろうか？もしできるだけ速くしたければ，答はイエスで，小さくなければならない．しかしこの問題を別の角度から扱うことができる．線上での遅延や，各素子の出力を別の素子に入力せねばならないなどの理由で，実際に使われているクロックの周期は 100 倍長い——すべてをそれほど速く変化させることはできないし，そうさせようとすると処理が混み合って動かなくなる．さて，反転器の速度を少しだけ遅くできないかというと，これは明らかではない——もし遅くすれば，速度の低下

*7 チャック・サイツとカルテクの共同研究者たちによる「ホット-クロック nMOS」についての 1985 年の論文は，「われわれはカルテクの同僚のアレイン・J. マーティンおよびリチャード・P. ファインマンとの「ホット-クロッキング」についての多くの興味深い議論を楽しんだとともに恩恵を受けた」という謝辞で終わっている．［編者］

に比例して計算全体にかかる時間が長くなるということは必ずしも正しくない．この点は不明確なので，$(Et)_{sw}$と表すわれわれの定数の値を正確に求めることは興味深い．

1つの求め方は，直接計算可能な特定のスイッチに対する定数の値を求めることである．そこで最も速いスイッチに焦点を合わせて，これに対する定数を計算しよう――どのスイッチを選んでもよい．まず基本的な式をまとめておこう．われわれのスイッチは，単一のトランジスターで，静電容量C_gを持ち，電圧V_gをかけるので，電荷は$Q=C_gV_g$である．そこでスイッチングのエネルギーは$(E)_{sw}=C_gV_g^2$である．さて$\tau=C_gR$なので，$(Et)_{sw}=C_g^2V_g^2R=Q^2R$，すなわち，スイッチを働かせるのに必要な電荷の2乗とスイッチを入れたときの最小抵抗との積になる．（この結果は，電流を考えて，電力損失からも理解できる．）そこで通常のトランジスターではこの散逸作用がいくつになるかに興味を持つのは当然である．われわれがこの定数を知りたい理由は，このようなデバイスを再設計して，エネルギー損失を減らすか時間を短くするかして，この定数を少し小さくできるかどうかを理解したいからである．

このどちらかというやさしい計算をするために，いくつかの物理的パラメーターが必要である．まず最初に，電子の電荷$e=1.6\times10^{-19}$クーロン，室温では$kT/e=1/40$ボルト，運動エネルギーの関係$(1/2)mv^2=(3/2)kT$（ここでmは電子の有効質量）を使うと「熱速度」v_{th}を定義でき，これはおよそ1.2×10^7 cm/sであることがわかる．薄くドープしたシリコン物質のいくつかの性質も必要である――電子キャリアーの表面チャネル移動度$\mu=800$ cm^2 V^{-1} s^{-1}と平均自由行程$l_{col}=5\times10^{-6}$ cm．前に行なったMOSFETでの解析のように，ゲートの下のシリコンの長さをL，幅をWと記す．1978年には，Lの典型的な値は6ミクロンであったが，1985年までに3ミクロンに下がっている*8．

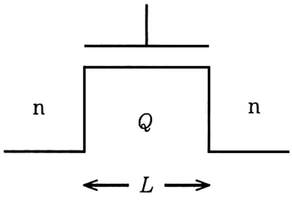


図7-34　簡単なMOSFET

ゲートの下で電子がパチャパチャと動いていて，その電子に力Fを時間τ_{col}だけ加えると仮定する．この時間を電子衝突間の平均時間にとるが，これはこの状況に対する物理学での当然の選択である．平均自由行程と衝突時間との間には直観的に満足できる関係$l_{col}=v_{th}\tau_{col}$がある．さて，標準的な力学によれば，この時間の最後に電子は運動量$mv_D=F\tau_{col}$を得る．ここで速度v_Dは力の向きへの「ドリフト速度」で，v_{th}にまったく独立である（そしてずっと小さい）．移動度は関係式$v_D=\mu F$によって定義されるので，$\mu=\tau_{col}/m$である．さて，ゲートの下を流れる電流をIとすると，$I=Q/$（ゲートの通過時間）$=Q/(L/v_D)=(Q/L)(\mu e)(V_{ds}/L)$が得られる．しかし，ソース-ドレイ

*8　1996年の標準的な技術では0.5ミクロンで，インテルのような主要製造業者からは0.35ミクロンのものが手に入る．［編者］

ン電圧は $V_{ds}=IR$ なので，抵抗として $R=L^2/(Q\mu e)=mL^2/(Qe\tau_{col})$ が得られる．（ついでに言えば，Si の中を動く電子の有効質量は自由質量の 10% 以下なので，m は自由質量とする．）さて，Q と R で表された散逸作用の式を使い，いまわれわれが導出した関係式を使うと，

$$(Et)_{sw} = N(L/l_{col})^2(3kT)\tau_{col} \quad (7.25)$$

が導かれる．ここでゲート下の(自由)電子数 N を導入した．$N=Q/e$ である．さて，式(7.25)の右辺の最後の2つの因子に焦点を合わせよう．$3kT$ はエネルギーで，電子の運動エネルギーのオーダーである．τ_{col} は時間で，衝突間の時間である．この2つの項の積**それ自身**を散逸作用──1回の衝突で散逸される作用──と定義すると，ここで何が起こっているかを理解する助けになるだろう．こう考えなければならないのではない．どうなるのかを知りたいだけである．この作用を $(Et)_{col}$ と呼ぼう．そうすると，次式を得る．

$$(Et)_{sw} = N(L/l_{col})^2(Et)_{col} \quad (7.26)$$

そこで，スイッチ全体に必要な Et が単一衝突の Et より2つの因子だけ大きいことがわかる．1つはゲートの下の電子数で，もう1つは平均自由行程に対するゲートの幅の比である．L を6ミクロンとし(したがって，L/l_{col} は約100になる)，ゲートの下の電子数 N を約 10^6 とすると，

$$(Et)_{sw} = 10^{10}(Et)_{col} \approx 10^{10}kT\tau_{col} \quad (7.27)$$

が導かれる．これは以前に引用したものと同じである．さて，これは**すさまじく大きい**量なので，事態をなんとか改善できないものかと希望する！　なぜこの数はこれほど大きいのだろうか？　われわれはこの数が決して基本的なエネルギー論での限界ではないことを第5章の考察から知っている．この数を少し減らすために何ができるのだろうか？

もちろん，ここまでの計算はすべて通常のシリコン VLSI 法に基づいたものである．そこで，われわれがすべきことはおそらくこの技術ではなく別の方法を調べることである．この問題をもっと一般的に，そしていくらか抽象的に考察しよう．君たちが誰かのために計算用デバイスの基本部品の素晴らしいスイッチを設計すると仮定しよう．これはスイッチングエネルギー $E_{\rm part}$ と対応するスイッチング時間 $t_{\rm part}$ を持つ．さてその人に多数のこの部品を与え，彼はそれらを使って回路を組み立て始める．しかし彼はこの組み立てをとても不合理で非効率的な方法で行なう．彼は次のように行なう(最初は少し抽象的に思われるかもしれないが，我慢しなさい)．まず彼は p 個のスイッチを並列に接続し，それら全部を同じ入力に接続する(図 7-35)．これらのスイッチはすべて同時に動作し，信号は左から右へ伝搬する．全部品がスイッチング時に散逸するエネルギーは明らかに $E_{\rm sys}=pE_{\rm part}$ で，それが起こる時間は $t_{\rm sys}=t_{\rm part}$ である．言い換えれば，$(Et)_{\rm sys}=p(Et)_{\rm part}$ である．これら全部が同じ答を与えることを考えると，こ

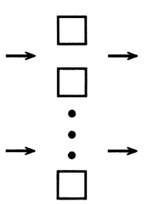


図 7-35　基本部品の可能な並列接続

れはばかげている．

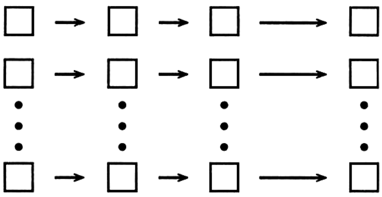


図 7-36　部品の直列接続

次に，その人はもっとばかげたことを行ない，これらの部品の組を直列に s 個接続する(図 7-36)．これは本当にばかげている！　直列になっている各スイッチは前のスイッチを反転するだけなので，彼の装置全体は単純なスイッチで，実際上は彼が始めたときに使った並列の装置と同じである！　しかしそれに比べると，君たちにはわかるはずだが，今度は $(Et)_{\rm sys}=ps^2(Et)_{\rm part}$ である．この結果はわれわれの問題にどのような関係があるのだろうか？　それは，1 回の電子衝突は，量 $(Et)_{\rm part}=3kT\tau_{col}$ に結びついた，**1 電子スイッチ**に似ていることである．このような衝突を基本的作用と考えることができる．ところでゲートの下で左右上下に急速に動いている電子は，互いにぶつかったり，ドリフトしたりして，同じことをしている．そこで，これらの電子は実際上は**並列に**作用していると考えることができる．並列部品数 p はゲートの下の電子数 N，つまり $p=N$ である．もちろん，1 つの衝突は電極間での電子の活動全体を説明するのに不十分である──実際の衝突数は平均すると L/l_{col} なので，直列との類推を使うと $(L/l_{col})\sim s$ である．そこでわれわれのエンジニアのばかげた手作業に基づいて $(Et)_{sw}$ に対するわれわれの結果を解釈できる．つまり $10^{10}=ps^2$ である！　確かに改善の余地がある？

うまくいった──それではこれをどのようにすれば改善**できる**だろうか？　まず部品を並列に置くことは完全にばかげたことなのだろうか？　決してそうでなく，精度のためには良いことである．きわめて敏感なので熱ゆらぎなどで間違った状態に変わってしまうような部品を使う場合がそのような場合である．そのような部品を並列に置き，出力を平均によって決めると，すなわち多数決で出力を決めると，系の信頼性は改善する．もし部品の誤動作の確率が 1/4 なら，400 個の部品を並列にすれば，系が間違った答を出力する確率は約 10^{18} 分の 1 である──すごい！　部品を**直列**に置いたらどうなるだろうか？　さて，私はこのことについていろいろ考えたが，まだ**何の**利点も見つけていない．直列にすることは信頼性を高めないだろう──遅延を増加させるだけである．私は $s=1$ 以外にする理由がわからない．

問題 7-3　われわれの電子モデルでは，$s=1$ は基本的な比 (L/l_{col}) を 1 に下げることに対応する．もしわれわれが実際にこの見解を真面目に受け取ると，興味深い問題が生じる．ゲートの下の電子の平均自由行程が**無限大**の最も極端な場合，言い換えると，電子が衝突しない場合を君たちに考えてもらいたい．そのようなデバイスの特性と動作を分析しなさい．確かに第一印象では，そのようなデバイスは決してスイッチとして機能できない──常に導通する．しかし，われわれは**慣性**を忘れていた．つまり導通するには，電子は速度を上げ，そして

速度を変えねばならないし，また速度がゼロでしか出発できない．そこで，とにかくゲートの下にはある電荷密度が存在する．事実，この完全な解析は，最初は真空管に対して平均自由行程が無限大の場合になされたが，これは確かにうまくいった．そのためこの種のスイッチが考案でき，解析できる──シリコンではこれができない(シリコンでは電子がある種の「蜜」の中を動いているように考えられる)．

一般に，どのような方法を用いてでも，平均自由行程を増加させLを減少させることにあらゆる努力をすべきである．$(Et)_{sw}$には100という因子がある(平均自由行程を変えるとτ_{col}も変わるので，10^4ではない)．現在のハードウェア設計はひどく評判が悪い！　エネルギー損失は莫大であり，スピードを上げると同時にエネルギー損失を減らせないという物理学的な理由はない──そこで君たちはそれを目指しなさい．君たちは自然ではなく自分の想像力に直面しているだけである．

すぐに思いつく提案は機械の大きさを単に縮小することである．この方法によって大きく進歩できる．Lを因子$\alpha<1$の割合で縮小しよう．つまり，$L\to\alpha L$となる．したがって，$(L/l_{col})\to\alpha^2(L/l_{col})$となる．ゲート下の電子数は面積と同じ割合で縮小するので，$N\to\alpha^2 N$となる．したがって，次の結果が得られる．

$$(Et)_{sw}\to\alpha^4(Et)_{sw} \qquad (7.28)$$

これは著しいスケーリング効果である．この性質はαのあまり小さな値までは信用できないが，この性質は部品を単に縮小させることが有利なことを示している．

私がここで提唱したEtという概念はこれらの事柄を考察する私自身の方法であり，間違っているかもしれない．(Et)が定数かもしれないというアイディアは量子力学の不確定性原理を思い出させるが，もしそれが本当だとしたら，私はその基本的な説明を知りたい．君たちが何かを気づくことができるかどうかを調べようとしてこの問題を研究する余地は確かにある．君たちがこれらのアイディアを批判したり議論したりできるものは何であっても価値がある．簡単な関係

$$\text{電力}=E/t=(Et)/t^2 \qquad (7.29)$$

は，散逸作用Etを下げることがもっと速い機械で電力損失を減らすことを示している．

7-3　VLSI の回路構造

ついに VLSI の基礎にある実際の物理的技術を議論することになった．トランジスターは実際にどのようにして作られるのだろうか？　とても大きなわれわれがそんなに小さなチップの上にこの全部をどのようにして載せるのだろうか？　その答は「とても，賢明に」である

──ただし基本的なアイディアは概念的にとても単純である．VLSIの製作手法全体は工業技術と製造業の偉業であり，そのすべてがどれほど素晴らしく見事なものであるかを普通の人が認識していないのは残念だ！　チップを作るのに必要な精度や技能はとても素晴らしい．人々はピンの頭の上にものが書けることを将来のことのように話すが，現在すでに可能なことに気がついていない！　われわれはいま容易に百科事典や聖書のような本の全部を，天使ではなく，ピンの頭のようなちっぽけな場所に書き込むことができる！　この節ではVLSI要素を作るために使われている基本的な過程を，かなり単純化した水準での分析で，調べよう．ここでもnMOS技術だけに焦点を合わせる．

7-3-1　プレナー製造工程技術

工程はすべて非常に純粋なシリコンの単結晶を使って始める．この物質は電子工学での応用が見つかる前から知られ研究されていたが，最初はシリコン単結晶はめずらしく，掘り出されたときは不純物だらけだった．現在では，実験室で，極度に純粋にすることができる．約4インチ四方*9で，奥行きの長いシリコンの単結晶を薄く輪切りにしてウェーハーを作る．この基板に集積回路を作るには，設計に従って必要な酸化物，ポリシリコン(多結晶シリコン)や金属をつけていき，ウェーハーをいくつもの層が重なった状態にする必要がある．前の議論でMOSFETのソースとドレインは，薄くp型にドープしたSi物質の上にn型領域を接合するのではなく，中にn型の不純物を注入したものであったことを思い出すこと──われわれが使っているシリコンウェーハーは実際にこのp型の物質であることを覚えておくことは重要である．どのようなことが起こっているのかを理解するために，最初の段階を少し詳しく説明する．

最初の段階は，最終的にはトランジスターのゲートの下の絶縁層を構成する役割を果たす非伝導性の酸化物層をシリコン上に作って巧みに処理することである．高温で，ウェーハー表面上に酸素を通すことで開始するが，これは酸化シリコン(SiO_2)の層を成長させることになる．この酸化物層は図7-37に示されている．次にこの酸化物を選択的な方法で取り除きたい．われわれはこれを非常に巧妙に行なう．酸化物の上に「レジスト(絶縁塗料)」の層を塗るが，これは有機物で，確実に密着するようにベーキングをする．このレジストの1つの性質は紫外線で破壊されることで，われわれはこの性質を使ってウェーハー上に電子回路の輪郭を刻み込む．テンプレート──「マスク」──をこの物質の上に置く．このマスクは紫外線に不透明な物質で上塗りされた透明な物質で，SiO_2を残したいチップの領域上を占拠する．(通常，マスクは領域全体にこのパターンを数多く繰り返し，多くのチップを1つのウェーハー上に作り出すことを可能にしている．これは後で別々のチップに切断される．) 次にウェーハーに紫外線(またはX

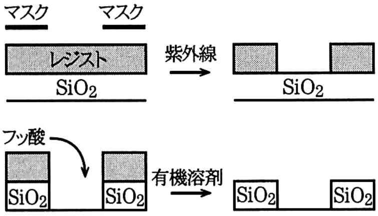


図7-37　チップ製造の最初の段階

*9　1996年には，直径12インチのシリコンの円柱が普通になっている．[編者]

線)を照射する．マスクの不透明な領域によって遮蔽されず，冒されたレジストは破壊され，洗い落される．これは SiO_2 の溝を露出させるが，この露出した SiO_2 を今度はフッ酸のような，強酸を使って取り除く．レジストの利点は──洗い落とされたものとは異なり──酸によって取り除かれないために，その下にある残しておきたい SiO_2 の層を保護することである．この段階の後では，上側にレジストの格子，その下に SiO_2 があり，その下にむき出しになった元のシリコンの格子がある．次に有機溶剤をウェーハーにかけてレジストを取り除き，その下の酸化物をそのまま残す．この結果は，「「シリコンの穴」がある酸化物層になる」と言うことができる(図 7-37)．

これはいくつかある段階のうちの最初の段階である．2 番目の段階は(たとえば，抵抗として使うために)回路の中で必要な任意の**デプレッション**型トランジスターの基本物質を置くことである．このトランジスターはエンハンスメント型のデバイスとは構成が異なり，ゲートの下でソースとドレインの間に薄い n 型 Si の層をもつ(図 7-38)．前に述べたように，$V_{th}<0$ なので，負電荷をゲートに置いてスイッチを開かないかぎり，そのようなトランジスターは永久に**閉じて**いて，電流は常に流れうる．先に進む前にチップ上にそのようなトランジスターを付けるための下地を置く必要がある．つまり，まずゲート領域の輪郭を描き，その次にこの範囲に n 型にドープされた Si の非常に薄い領域を作り出す必要がある．このために，もう一度ウェーハーをレジストで覆い，透明なところが空乏領域を表すマスクをその上に置く．もう一度，ウェーハーを紫外線または X 線で照射し，今回はウェーハーをレジストで覆われたままの状態にしておく．レジストの間には露出したシリコン基板の箇所が点在している．この開いた領域に，必要な空乏領域を作るために，リン，ヒ素，またはアンチモンをドープする．レジストはこれらのイオンがシリコンの残りの部分に入り込むのを防いでいる．これが終わると残っているレジストを洗い落とす．

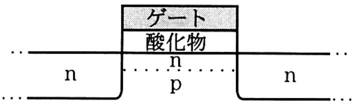


図 7-38　デプレッション型トランジスター

次に作る層は**ポリシリコン**(多結晶シリコン)層である．高濃度にドープしたポリシリコンは，金属ほどではないが，よく伝導し，とりわけトランジスターのゲートを作るのに使われる．このゲートは絶縁性酸化物の薄い層で基板から分離されているので(図 7-38)，最初に行なったようにウェーハーを酸化物の別の薄い層で覆わねばならない．前のように，ウェーハーを酸素の中で加熱する(これはウェーハーの酸化物の深さを不均一にすることに注意)．その後でウェーハーをポリシリコンで覆い別のマスクを上に置く──今度は不必要なポリシリコンを取り除けるように設計されている．この後にトランジスターのドレインとソース(そして一般に拡散層)を作らねばならない．つまり，残ったシリコンのすべてを適切に(すなわち，リンを使って)ドープせねばならない．ポリシリコンの下以外のすべての酸化物を取り除き，露出した Si 領域に大量のドーピングをすることによってこれを実行

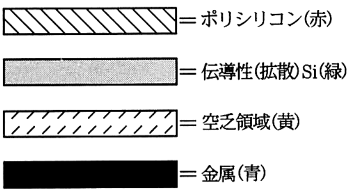


図 7-39　チップ経路の規約

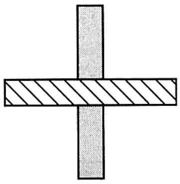


図 7-40　エンハンスメント型トランジスターの概念図

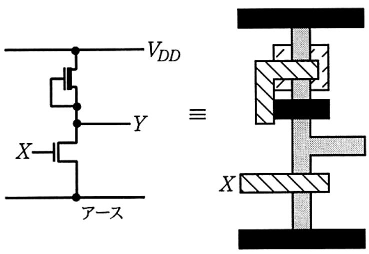


図 7-41　完全な反転器

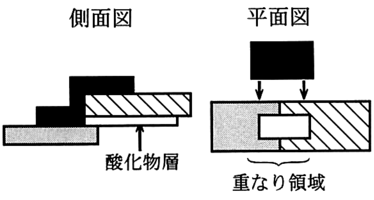


図 7-42　ポリシリコン-拡散層の接触

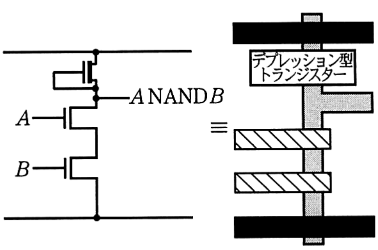


図 7-43　NAND ゲート

する．ポリシリコンの下の空乏層は保護されているのでそれ以上汚染されることはない．

さて，われわれはこの工程でエンハンスメント型トランジスターをどのように作れるかも理解できる．図を描くために，図 7-39 に示したいろいろな層の規約を採用する(最も一般的な規約は色で塗り分けられている)．もう1つの層の金属層[*10]をここに加えた．この層は電流をかなり長距離まで運ぶのにポリシリコンや拡散層より優先的に使われる「平らな電線」を構成している．(電源は通常は金属経路から引き出される．) また，層の間で電流が要求通りに自由に流れるように接点を付けることも必要である．この規約を使うと，エンハンスメント型トランジスターを図 7-40 のように描くことができる．このトランジスターはポリシリコン経路と拡散経路との交差点にすぎない！もちろん，2つの経路は物理的に接触しているという意味では交差していない――これらの間には絶縁性酸化物層がある．

完全な反転器には直列の抵抗が必要である．前に議論したように，抵抗としてデプレッション型トランジスターを使う．反転器回路を図 7-41 に示す．この図には両方とも金属経路の電源とアース線が入っていることに君たちは気づくだろう．製造過程では金属経路と交差する拡散経路の部分を露出させておくことが，電気的接触を保証するために必要である．このような特徴は平面図からはわからない．(実際の回路は図 7-41 のように全体が平坦にはレイアウトされていない，つまり，うまくすき間なく詰めた小さな箱のなかで，重なってできている．詳細についてはミードとコンウェイの本を参照．) たとえば，もしあるトランジスターのソース，またはドレインを別のゲートへの入力として使いたければ，同じような処置が必要で，拡散経路をポリシリコン経路に接続しなければならない．明らかに，ある種の直接接触が必要である．そうでなければ，線が交差する場所にキャパシターやトランジスターができてしまう．図 7-42 に示されているように，金属を直接的に拡散/ポリ層に重ねて接触させる，いわゆる「隣接コンタクト」の方法を使うことができる．

もっと入り組んだ論理装置の例を示すために，NAND ゲートを考察しよう．これを作るために行なわねばならないことは，前の回路を使い，そして拡散経路ともう1つのポリシリコン経路とを交差させて，もう1つのトランジスターを作ることである(図 7-43)．この回路では2つのトランジスターのところでポリシリコン経路が拡散経路を少し越えて延びていることに注意しなさい．なぜそうなっているのだろうか？　それは，チップ上のいろいろな経路を互いにどのように配置するか，経路をどれくらい大きくしなければならないかなどについて正確に決める多くの設計規則があるからであるが，ここではそのいく

＊10　1996 年現在では3つの金属層が代表的で，専門家には5つが利用可能である．これらの「金属」層のうちの代表的なものの1つはポリシリコンであろう．［編者］

つかを簡潔に列挙する．(この「ラムダベース」の設計規則のもっと完全な説明についてはミードとコンウェイの本を参照.) まず長さの単位λを定義し，チップ上のすべての長さをこの変数を使って表そう．λは1978年には約3ミクロンであったが，1985年までに1ミクロンに低下し，そして時とともにさらに低下している．拡散経路とポリシリコン経路に対する最小幅は2λである．金属の導線の幅は少なくとも3λなければならない．これは「電気泳動」と呼ばれている金属原子が電子流方向にドリフトする現象の可能性を阻止するためである．もし導線が特に細ければこの現象は重大な破壊になりうる(図7-44)．これらは最小値であり，もし望めば経路をもっと幅広くできる．もう一組の規則は導線をどれほど近づけて並べられるかに関係している．伝導経路は，電流が経路間を横に流れてしまう電圧の絶縁破壊の危険性のために，互いにあまり近づけて設置できない(図7-45).

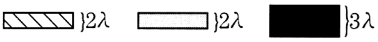


図7-44 シリコンチップの経路の幅

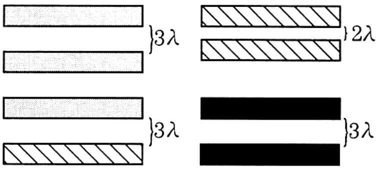


図7-45 シリコンチップの経路の分離

金属経路は接触をとらずにポリ経路および拡散経路の上に設置できる．すでに述べたように，ポリ経路が拡散経路と交差する場所にはトランジスターがある．そのようなデバイスでは，ドレインとソースとが短絡した結果としてゲートのまわりに伝導通路が形成されるのを妨げるために，ゲートを構成するポリの線が拡散領域の端を越えて延びていることが重要である．製造誤差を考慮に入れて，通常は少なくとも2λのはみ出しを要求する(図7-46).

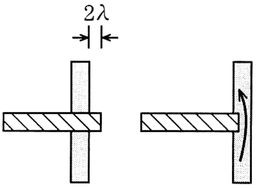


図7-46 トランジスターに対する規則

層間の接続も考えねばならない．金属線を別の経路につなぐときには，接触を確実に良くしなければならない(接触部は通常は正方形に作られている)．接触を保証するために，金属と経路とを，ただ面と面が接触するように置くだけでなく，金属を通って周辺へ漏れるのを妨げるために，接触部の周囲に少なくとも経路物質から距離λをとらねばならない．ポリの線への接続であろうと，拡散の線への接続であろうと，金属の線への接続であろうとこの規則は正しい(図7-47).

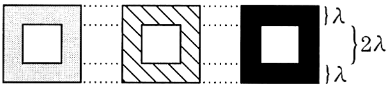


図7-47 接触についての規則

7-3-2 回路設計とパス・トランジスター

実際に特定の回路を作るためには，必要な(通常は途方もなく複雑な)マスクをすべて設計し，それを製造業者に送る．製造業者は，この製品を供給するために前に記述した製造過程でそれらを作る．回路の図を描くための標準的で役立つ方法がある．この図は配置のトポロジーを示すもので，回路の幾何学を示すものではない．すなわち，どの経路が何で作られているのか，そして何がどこに接続されているのかなどを示すが，縮尺，つまり経路の適切な長さなどは示さない．たとえば，NANDゲートの図(いわゆる「スティック線図」)を図7-48に示す(この中には各タイプの経路に対して採用する新しい線の規約も示した).

この図は回路の重要な相互結線のすべてを示しているが，完成した製品を実際に調べると，各部分の実際の大きさは極端に違っているだろう．ある特定の回路を考察するときには，この事実はわれわれに関

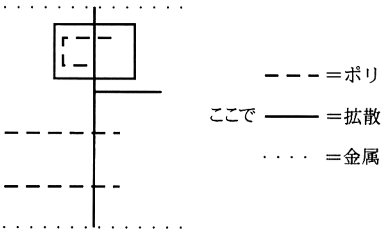


図7-48 NANDゲートの「スティック線図」

係がないので，以下ではスティック線図を採用する．物事をもっと簡単にするために，ある種の「中途半端な」略記法をときどき使う．この略記法ではチップ上の部分的な回路をブラックボックスで表す(よく使われる方法)．たとえば一連の反転器があれば，すべてのトランジスターのスティック線図を繰り返して描くよりも図 7-49 のような図を使う方が簡単である．ここで小丸の付いた三角形は反転器の慣習的な記号で，線の規約は図 7-48 に説明されている．

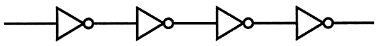


図 7-49　一連の反転器の単純化された回路図

よくある型の回路はシフトレジスターである．図 7-50 ではこれをポリシリコン経路が交差している二重クロック付きの一連の反転器として表す．2 つの(相補的な)クロックパルスはポリシリコン線で送られる．ポリシリコン線が拡散線と交差するところでは，いわゆる**パス・トランジスター**が形成される．この名前の由来は，ゲートに順方向にバイアスをかけた場合に電流をソースからドレインへ(すなわち，図 7-50 で左から右へ)流すことだけを行なうためである．ポリシリコン線に対するクロックパルスがオンのときはいつでもこれが起こる．次のパルスでは次の反転器が切り替わり，次のクロックパルスまでその新しい値を保持する．図 7-50 がどのように動作するのかを理解するために，読者は第 2 章のクロック付きレジスターの議論を読み返しなさい．これはフリップ-フロップと論理ゲートの集まりよりも簡単で，より利用しやすい配列である．ついでだが，これをメモリー格納庫として使いたければ，このような配列を閉じる(すなわち，配列を「円形に」する)ことができることにも注意しなさい．

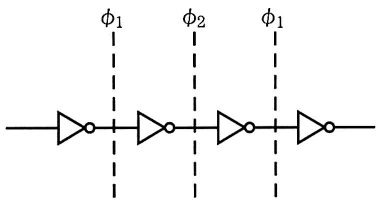


図 7-50　シフトレジスター

7-3-3　プログラム可能なロジックアレイ

プログラム可能なロジックアレイ(PLA)を使って，機械の **if-then 制御**の問題──すなわち，ある 1 組の入力データが与えられたとき，機械が次に行なうことをどのようにして決めるのかという問題──を調べるときがきた．たとえば，「もしこれとこれがゼロならば停止せよ」とか「両方のビットが 1 ならば 1 を桁上げせよ」というような問題である．抽象的には，われわれが次に行なうべきことを示す，機械のある部分から出てくる情報がある．この情報はある「センサー」(私自身の言葉で，技術用語ではない)に行き当たるが，このセンサーはわれわれに現在の状態を教えてくれる．いったんこれを知れば，たとえば，加算器に加えたり引いたりすることを告げて，これを実行できる．この命令，あるいはもっと一般に，1 組の命令は 1 組の線から出力するデータの形をとる(図 7-51)．

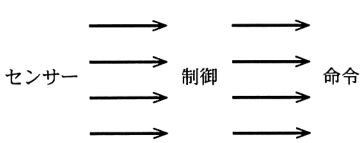


図 7-51　一般的な制御デバイス

これを実行するデバイスの設計の第一段階は，明らかに，ある与えられたセンサーの組に結びついている命令の組が何であるかを知ることである．これはかなり容易である．たとえば，命令を表 7-1 のように表せる．

これは次のようなことを意味する．表の左側の各行はセンサー線上

表7-1　制御デバイスに対する命令の組の例

センサーの線					命　令					
1	2	3	4	5	a	b	c	d	e	f
1	1	0	1	0	1	0	1	1	0	1
1	0	0	1	1	1	0	1	1	0	1
0	1	1	0	1	0	1	1	0	1	0
		⋮					⋮			

のビットのある配列(この例では線は5本)を表している．左側のセンサーの組が与えられたとき，これに対応する右側の行は命令線へ送り出すビット(この場合，6つ)を表す．この表の右側の1は「もしこの線からの入力が1ならばあることをしなさい」——たとえば「加えなさい」とか「点灯しなさい」——を意味し，その一方，0は何もしないか，または他のあること，たとえば「状態Xをそのままにする」とか「消灯しなさい」とかを行なうことを意味する．制御系を作るきわめて直接的だが非効率な方法は単にこの表を，センサー線はメモリーアドレスとして，制御線はこのアドレスの内容として，メモリーに格納することである．このようにわれわれはセンサー線のすべての可能性のある組み合わせに対して実行すべき操作を別々に格納する．このメモリーの内容は固定されるべきものであり，すべてを読み出し専用記憶(ROM)デバイスに格納した方がよいだろう．

このわかりやすい処理法に起こりうる唯一の障害はタイミングから生じる．つまり，ある命令が他の命令より前にROMから出ていき，機械の状態を変化させ，センサーを混乱させることが想像される．この効果が前のセンサーの組の取り扱いが完了する前にROMにフィードバックされるかもしれない．もしこれが起こったらかなりひどいことになるだろう．しかし(君たちは私よりずっと良く知っているはずだが)命令の検索と利用が異なる時刻に行なわれることを保証するためにROMの両端にクロック同期のレジスターを配置することによって，これは普通は避けられる(図7-52)．ϕ_1がオンのとき，センサー線はROMへ入力し，ROMは対応する制御信号を調べる．この制御信号は，ϕ_2がオフなので，外へ出られない．すべてが確実に落ち着いたとき，つまりすべてのセンサーの情報が入り命令の組が選ばれたことが確実なときにのみスイッチϕ_2はオンになる．ϕ_2がオンのときは，ϕ_1はROM入力を凍結するためにオフになっている．この外部クロックがオンだと，命令の組は出ていくことができ，ROM入力に影響を及ぼさずに機械のその他の部分に到達する．そしてこの過程がつづく．

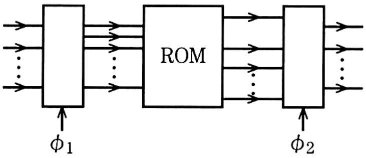


図7-52　クロック同期のROM制御系

このように，制御はとても，とても単純にできることがわかる．しかし，われわれはそれを効率的にもしたいのでこの種の方法に不満である！　一般に，メモリーに2^nの項目を詰め込むことはいくらか浪

費的である．たとえば，しばしば 2 つ以上の入力状態が同じ出力状態になったり，いくつかのセンサー線が ROM に入る前にいつも多入力 OR ゲートでフィルターにかけられることがある．したがって，この表にはきわめて多くの余分な情報が含まれている．そこでわれわれは ROM を完全に取り除き，基本に立ちもどって，多数の論理ゲートを含む回路を開発しようという気になる．これは初期に行なわれた方法であり，使用する論理ゲート数を最小にするために定理を利用したりして，非常に複雑な論理回路を注意深く組み立てる方法で，ROM は使わない．しかし，最近の回路はとても複雑で，人間の頭の能力には限界があるので，ROM 法を使う必要がしばしばある．しかし可能性のある出力数が十分に少なく，論理回路だけを使うとずっとコンパクトな装置ができるので ROM を必要としないという中間的な場合がある――その構成は**あまり**複雑でないので設計可能である．そのような事例を説明するために，われわれが第 2 章で最初に出会った，いわゆる「プログラム可能なロジックアレイ(PLA)」を調べる．これはセンサーの入力が送り込まれ，要求された命令の組を出力する論理ゲートの整然とした配列である．理想的な場合には，そのような配列には冗長性がない．「ブラックボックス」図を使うと，一般の PLA は図 7-53 に示す形を持つ．

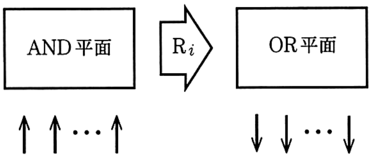


図 7-53　一般の PLA

この図からわかるように，PLA は 2 つの主要な部分，つまり AND ゲートだけから作られる「AND 平面」と OR だけから作られる「OR 平面」から構成されている．2 つの平面は導線のブリッジ R で接続されている．入力信号は AND 平面に入り，処理されて R 導線を通って OR 平面に入る．そこでさらに進んだレベルの処理がなされ，信号が OR 平面からの出力として出てくる．この出力はある特定の入力に対応した「次は何」命令の組になっている．

3 本の入力線 A, B, C と，4 本の出力線 $Z_1, \cdots, Z_4$ がある場合を考えよう．各入力信号は AND 平面へ送り込まれる前に 2 つの部分，つまりそれ自身とその補数，たとえば，A と NOT A に分けられる．われわれはいま，各信号を NOT, AND および OR で処理できるデバイスを持っている．言い換えれば，このデバイスは任意の論理関数を何でも表すことができる．実際にそのような配列のトランジスター構造を示すために，特定の PLA を選ぼう．機械のある部分の状態を示す 3 つの入力と 4 つの可能な出力がある――この出力は発射されて，機械に対して次に何をするのかを告げるパルスである．さて出力 Z が次のような入力のブール関数だとしよう（$\vee$ = OR ; $\wedge$ = AND ; $'$ = NOT).

$$\begin{aligned} Z_1 &= A \\ Z_2 &= A \vee (A' \wedge B' \wedge C) \\ Z_3 &= B' \wedge C' \\ Z_4 &= (A' \wedge B' \wedge C) \vee (A' \wedge B \wedge C') \end{aligned} \tag{7.30}$$

Z を計算するために必要な A, B, C の個々のブール関数は一連の AND とそれにつづく OR の積として書けることはすぐには明白でない．しかし，第 2 章で一般の論理関数について示したように，実際にはそうなる．この例では，AND 平面からの計算できる出力 R_i は A, B, C の AND と NOT のみを含んでいなければならない．したがって，R_i を

$$\begin{aligned} R_1 &= A, \\ R_2 &= B' \wedge C' \\ R_3 &= A' \wedge B' \wedge C \\ R_4 &= A' \wedge B \wedge C' \end{aligned} \tag{7.31}$$

と定義すると，Z 出力をこれらの R についての OR 演算(または恒等演算)のみで書けることが容易にわかる．

$$\begin{aligned} Z_1 &= R_1 \\ Z_2 &= R_1 \vee R_3 \\ Z_3 &= R_2 \\ Z_4 &= R_3 \vee R_4 \end{aligned} \tag{7.32}$$

すべてのブール関数がこのように因子分解できることは一般的な結果である．この関数に対する PLA を図 7-54 に示す．

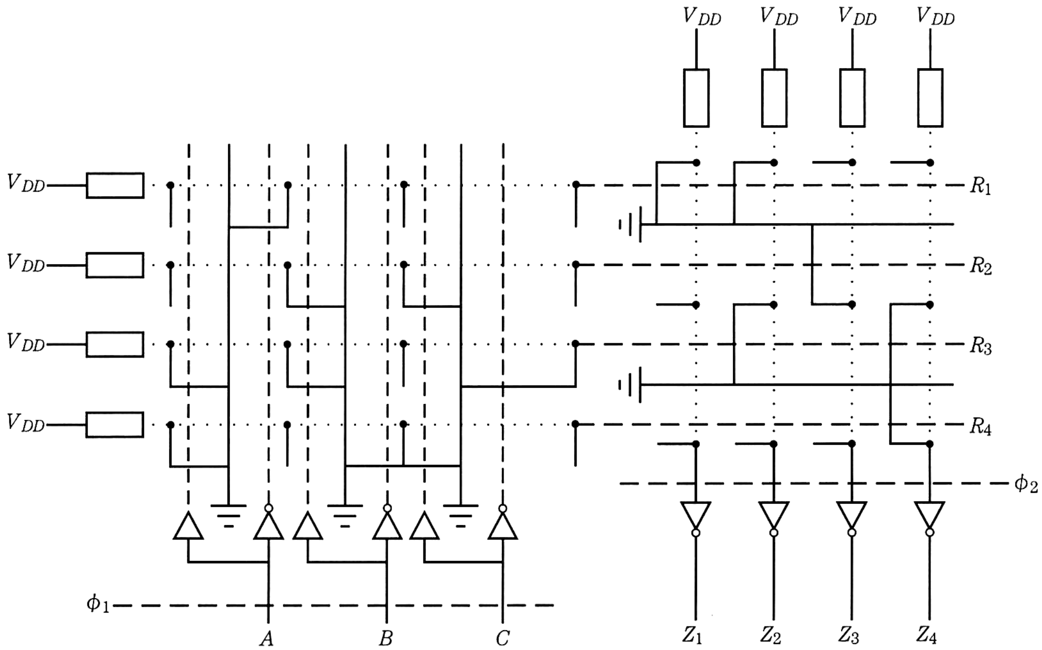


図 7-54　PLA の回路図

この回路がどのようにして求めていた変換をするのかを電子レベルで解くことは君たちに任せる！

一般に，PLA の構造の 90% 程度は実際の関数に無関係である．その結果，PLA は通常は標準の設計にいくつかの回路を付け加えて構成される．たとえば，図 7-54 の回路は汎用の AND 平面を利用して，

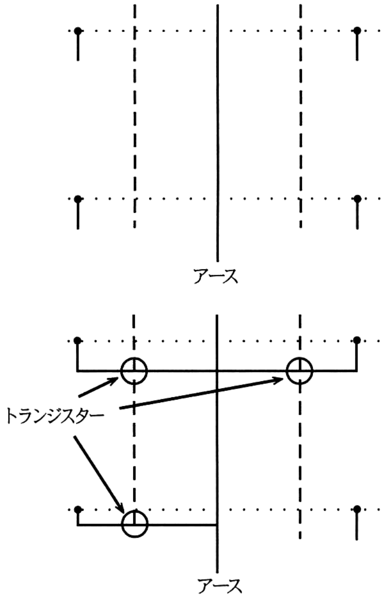


図7-55　汎用の AND 平面と修正した形

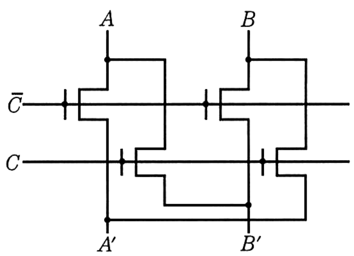


図7-56　交換回路

それにいくつかの拡散経路をうまく適切な場所に賢明に付加することによって，われわれが望むような回路に変えた結果である(図7-55).

これは非常に実用的な方法である．もちろん，もっと多くの線が必要なら製造業者のカタログでどの基本配線が利用できるかを調べねばならない．ついでだが，一般の OR 平面が本質的に AND 平面を 90 度回転したものだということを図7-54から読みとりなさい．

問題7-4　さて，君たちが解くべき面白い問題を出そう．これは現実のデバイス設計のときに実際に起こった問題である．問題はこうである．われわれは2本の線 A と B を制御線 C で切り替えたい，すなわち交換したい．C とその補数 $\overline{C}$ が与えられていて――それらはどこかからか入ってくるが，われわれはそれがどこかについては気にしない．そしてもし制御 C がホットなら A と B は場所を換え，もし C が冷たければ場所を換えない．これはわれわれがよく知っている制御付交換の変形である．われわれが使う回路図は，図7-56のようなものである．

規則を繰り返すと，$C=0 \Rightarrow A'=A$，$B'=B$; $C=1 \Rightarrow A'=B$，$B'=A$．君たちはすべてがどのように動作するかがわかるはずである．私が君たちにして欲しいのは次のことである．

(a)拡散，ポリ，そして金属の正しい規約を使ってスティック線図を描きなさい(ヒント：入力 A と B は金属線で入る).

(b) λ 設計規則に従った正式のレイアウトをグラフ用紙に描きなさい．

(c)この回路はその回路構造を単に繰り返し(C, NOT C 線を拡張すると)もっと多くの $A, B, \cdots$ などを入力をさせられるように容易に修正できる．さて8つの入力の対が上から入るとしよう．各対に対して水平に 14λ と，境界に余分に16または 20λ があるので，約 132λ の全幅が利用可能である．しかし，150λ の奥行きが許されている．さて，A と B などが回路から出ていく線も金属にしたい．この回路を設計できるか？　もし必要なら左側からもっと多くの C が入ると仮定してよい．

7-4　機械設計についてのさらなる制限

どのようなコンピューターでも最も重要な部品の1つは**導線**であるということはそれほど考えなくても理解できる．われわれは導線を――もっと一般にはポリシリコン経路を含む伝送経路を――われわれの機械の設計に必要な方法に対して影響を及ぼすような物理的性質を持つ，現実の物理的物体であることを忘れて，ある理想化された方法でこれを取り扱うことに慣れてきた．この最後の節では，導線が機械の設計に重要な役割を果たす2つの様式を考察したい．第一は導線の長さがクロッキングをどのようにして台無しにしうるかという，いわゆる「クロックスキュー」問題に関係している．第二はもっと簡単な問

題，つまり導線が場所をふさぐという事実，そしてコンピューターを組み立てるときには導線を取り入れるのに十分な余地を確実に残すようにしておいた方がよいということに関係している．

7-4-1　クロックスキュー

一般の PLA をクロックに同期させる議論にもどろう．図 7-57 のような一般の形の 2 つのクロックパルス ϕ_1 と ϕ_2 を使ったことを思い出しなさい．アイディアは ϕ_1 がオンになっている間に PLA にデータを送り込み，それからしばらく論理ゲートが仕事に取りかかり，出力を準備するようにさせるなどの処理をするということである．これが遅延時間を導入し，2 つのクロックを単純に相補的にしない理由である．それから ϕ_2 をオンに切りかえ，この間にデータを出力させる．これはすべてとても簡単で単純に思われる．しかし，現実の機械では，問題がありうる．まず第一に，回路素子のゲートの充電にゼロではない時間がかかり，これが遅延や時間のずれを導入する．また，クロックの信号は金属やポリシリコンなどの**導線**で送られる電流のパルスで，**これらのパルスの伝達には有限の時間がかかる**．短い導線で送られるクロックパルスは長い導線で送られるパルスより前に終端に届くだろう．われわれは，単純な導線を，図 7-58 に示すような構成部分を一列に無限に接続するという興味深い方法でモデル化できる(有限な場合には一連のパス・トランジスターのモデル化と考えられる)．

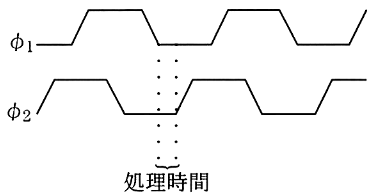


図 7-57　PLA のクロックパルス

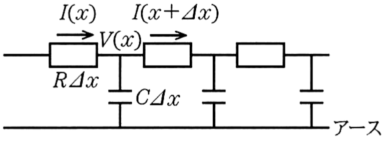


図 7-58　単純な導線の無限極限模型

キャパシターを途中に挟んでいる一連の抵抗器がある．もし無限に多くの小さなキャパシターや抵抗器が無限に束になっているとすれば，実質的に単位長さあたりの抵抗 R と単位長さあたりの静電容量 C を持つ導線があることになる．さてわれわれがしたいことは，線(金属である必要はなく，ポリシリコンでもよい)の一端に信号を入れて，その信号が他端へ伝搬するのを待つことである．原点から導線に沿った距離を x とする．各接合点で電位 $V(x)$ とそこに流入する電流 $I(x)$ を定義できる．$\delta x\to 0$ の極限をとると，初等数学と電気学によって一組の方程式

$$\partial I/\partial x = -C\partial V/\partial t \qquad (7.33)$$

$$\partial V/\partial x = -IR \qquad (7.34)$$

$$\partial^2 V/\partial^2 x = RC\partial V/\partial t = \tau\partial V/\partial t \qquad (7.35)$$

が得られる．ここで，$\tau=RC$ と置いた．式(7.35)は**拡散方程式**の例である．電荷は一端に流れ込み，この系の中を拡散する．グリーン関数で表されたこの方程式の解の一般形はよく知られている．われわれの場合の境界条件を使うと，解は

$$V(x,t) \propto \exp(-x^2\tau/4t) \qquad (7.36)$$

である．もし導線の全長が X なら，導線に信号を載せる時間は X^2 に比例することがこの式から容易にわかる．1 mm のポリシリコンの場合，この時間は 100 ナノ秒になる．2 mm の場合は 400 ナノ秒になる．特に，君が信号を載せる時間が距離に比例する伝送線を使い慣れ

ていれば，これはかなりみじめな線である．しかし，金属は低抵抗なので，信号を載せる時間は比較的短い．そこで，信号をとても遠くに送りたければ，金属で送るべきである．

クロックに同期させる問題は計算を行なうためにとても重要なので，別の手段，つまり情報の流れを制御する別の手段を探るように奨励されるのは当然である(実は，私がこれについて少ししか話さなかったので，君たちは大切でないと思うかもしれないが，本当はとても重要である)．いわゆる同期されたクロッキング――われわれがこれまで考えてきた唯一の型――という標準的な方法の問題点は，機械の設計で，系の各部分において「最悪の場合のシナリオ」を考慮しなければならないことである．出力を出すのに，たとえば t 時間単位ないし，$5t$ 時間単位かかる複雑な加算器から出力をとらねばならないと仮定しよう．そうすると，たとえ t 時間単位後に出力が勢いよく出てきたとしても，遅く出力するというとても可能性のないことのために少なくとも $5t$ の間はオンしていなければならない．このことから，ひどい時間の非能率となる可能性がある．ところで，機械を設計するもう1つの方法は，まだ市販されていないが，「非同期法」である．この方法では**加算器にタイミングを制御させる**，つまり，準備ができたら教えるようにさせる！　計算を実行し，データを送る準備ができたという信号を送らせるのである．この方法ではタイミングは計算素子自身によって制御され，外部のクロックの組によっては制御されない．

興味深いことに，少し考えると，同期系にも解決すべき非同期の問題があることがわかる．たとえば，そのような機械がキーボードからデータを受理したり，別の機械を接続しなければならないとしたら，何が起こるかを考えよう．キーボードはデータを送る「正しい時間」について何も知らない．われわれは**バッファ**，つまり機械のクロックが正しい状態にあるときだけデータを機械に入れるようにする小さな箱を持たねばならない．データがあまりに遅く入ってきたときに，データをいま受理するか，または次のサイクルまで待つかという決定をしなければならない．決定しなければならないという事実は，バッファが決定できない**ちょうど**そのときに入ってくるデータによって，機械が動かなくなるという理論的な可能性のあることを示唆する．これは魅惑的な問題で，よく考えてみる価値がある．

7-4-2　導線の詰め込み：レントの規則

われわれは今まで，トランジスター，VLSI 等々について議論してきた．そして，これが機械設計のハードな部分だと考えている．しかし大きな設計が終わり，組み立て始めるときはいつでも，君たちが解決してきたすべてのアルゴリズムなどでは十分でないことを発見する――常にあるものが邪魔をすることがわかる．そのあるものとは，**導**

線である．導線について次に考察する．

導線は系の設計における現実的問題であることを強調したい．われわれは導線が引き起こす1つの困難，つまり導線に載せるのに要する時間が有限である結果として生じるタイミングの問題について議論してきた．もう1つの問題は，チップ同士を接続*11するための配線に必要な空間はトランジスターのような機能部品の接続に必要な空間より大きいことである！　さて，導線が至高のものとして永久に君臨する保証はない．たとえば，光ファイバーを使い，異なる周波数の光を利用すれば，多重通信を1本の線で送れる．人々は時折挫折して，放送システムとの類推で，コンピューターを組み立てるというような華麗なアイディアの夢を見始める．その機械では各構成部分が特定の色の光を(たとえばLEDによって)放出し，その光は機械全体に送信され，周波数に感度のある構成部分が受信して作動するというような夢である．しかし，現時点では電流の伝送に広く使われている方法は導線によるものであり，私は導線についてしばらく議論して時間を費やしたい．特に，一般的設計で必要な導線の量は**どのくらい**かという問題について話したい．

さて，ここで導線の詰め込みについて私が話せることはほとんどない──結局，導線は導線にすぎない──．しかし，**レントの規則**という経験則があることがわかる．この規則は，この問題に光を当てると称している．これは奇妙な規則であり，私はこの規則が一般にどのくらい正確なのかについて実際に保証できないが，IBMでの**経験**に基づいた実情らしく思われる．この規則は次のようである．回路ボードのようなユニットがあると仮定し，ユニットを部品の入った大き過ぎもせず小さ過ぎることもない「セル」に分割する．これらのセルは，たとえば，個々のチップである．ここで次の仮定をする．

(1)各セルにはt本のピン，つまり端子がある

(2)N個のセルはユニットを作る

(3)ユニット上の端子の数，つまり出力のピンの数はTである

言うまでもなく，これらの数にはあるゆとりの幅があると解釈されねばならない．われわれは，全部品が多く交信するようにすべてを接続しようとすると仮定する．また，詰め込みによって導線の長さを最小にしようとすると仮定する．そうすると，レントの規則によれば

$$T = tN^r \tag{7.37}$$

である．ここで$0.65 \leqq r \leqq 0.70$（この不等式は近似にすぎないので$r = 2/3$ととる)．言い換えれば，この規則はユニットに出入りする導線の数($\sim T$)とユニットに詰め込まれているセルの密度($=N$)とを関係

*11　私はいまはチップ上の「導線」についてそれほど関心がなく，チップを互いに接続すること，つまりチップを接近して積み重ねるのに妨げになる導線の実際の束などに関心を持っている．[RPF]

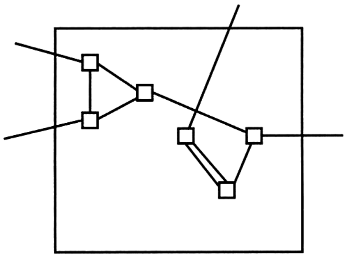


図7-59　ボード上の基本的なセルの図による描写

づけていると主張している．率直な最初の疑問は，「なぜ $T=tN$ ではないのか？」だろう．それは，多くの導線がユニットの**内部**を結ぶ線であるというわかりきった理由のためである(図7-59)．

ユニットとセルの階層構造を「上」に移動することによって，式(7.37)のような式がどのようにして生じるのかを理解できる．われわれはセルが互いに結合されたユニットを考えてきた．ここでは互いに結合された**ユニット**を考える．そこでもっと大きなユニットの「スーパーユニット」があると想像しよう．このスーパーユニットのセルは元のセルを含んでいるユニットである．このスーパーユニットは M 個のユニットを含んでいると仮定する．ところで，われわれは解析の基本的階層を設定していないので，これらの2つの状況間のスケーリングには整合性がなければならない．スーパーユニットの端子数を T_S 本としよう．明らかに M 個のユニットの各々には T 本の端子がある．その結果，レントの規則によって，

$$T_S = TM^r \tag{7.38}$$

である．しかし，解析の最初の階層にもどると，スーパーユニットを，各々が t 本のピンを持つ NM 個の元のセルから構成されたものとして取り扱える．そこでレントの規則を使うと，次式が得られる．

$$T_S = t(NM)^r \tag{7.39}$$

式(7.37)を使うと，式(7.38)と式(7.39)は明らかに一致することがわかるので，レントの規則は正しいスケーリング性を持つ．このことは非常に重要である．

この取り扱いは r の値については何も教えていないことに注意しなさい(規則の形と式(7.37)の後につづく議論から，r が1未満でなければならないことは明らかであるが)．この指数はどこから来るのだろうか？　さて，この選ばれた値は経験から導かれたものであり，この経験は論理回路の設計と結合における幾何学の問題に影響を受けたはずであるということを記憶しておくべきである．すなわち，r の値に対してあるすっきりした論理的理由があり，見事な数学的取り扱いの結果であると考えたくなるかもしれないが，この規則は通常の設計方法の産物である可能性がある．しかし，しばらくの間は，この注意を留意しつつ，この規則が一般の場合に**成り立つ**と仮定し，この規則が導線の詰め込みについてわれわれに何を教えるかを調べよう．

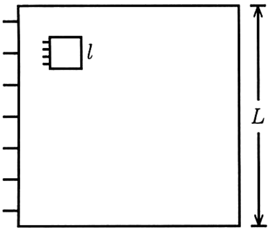


図7-60　一般の2次元ユニット

2次元の場合にもどろう．辺の長さがたとえば L cm の正方形のボードがあると仮定し，これをユニットとしよう．そこに，各々の長さが l cm のセルを詰め込む．そこで，ボード上のセルの数は $N=(L/l)^2$ と書ける(図7-60)．さて，各セルにどのくらい多くの端子を取り付けられるか──ただ単に端子を互いに近づけて配置できるか──について制限があると仮定する．セルのへりの上にあるピンの1 cm あたりの最大数が s_C だとしよう．ボードの端子に対してもピンの最

小間隔があると仮定し，へりの 1 cm あたりのピンの最大数を s_B とする．その結果，レントの規則は次のようになる．

$$T=(4s_BL)=tN^r=(4s_Cl)(L/l)^{2r} \tag{7.40}$$

そして次式が得られる．

$$s_B=s_C(L/l)^{2r-1} \tag{7.41}$$

この式から，もし $r>1/2$ なら，L を増加させるのにつれて，内部にあるいろいろながらくたを処理するために，ボードのへりの 1 インチあたりにますます多くのピンが必要になる．したがって，最終的にはピンがぎっしり詰まってしまうだろう．そのため機械を大きくしていくと，配線の問題がますます重大になる．この話の核心は，周囲の長さは面積の平方根に比例するが端子の数は，(レントによれば)ずっと早いスケーリング因子の，2/3 乗に比例するという事実である．多くの導線が圧縮できない混乱した状況になることは不可避で，われわれは場所を作るために，セルの間隔を広げ，もっと多くのボードを使い，そしてボードの間隔を広げるなどしなければならない．

さて，とても興味深いことに，この議論を 2 次元ではなく，3 次元でし直すと，異なる結果が得られる．3 次元ではへりの長さを表面積[(長さ)^2]で置き換え，面積を体積[(長さ)^3]で置き換える．明らかに，面積は体積の 2/3 乗に比例し，端子の数と同じになる！　そのために 3 次元ではまさにうまくいく──われわれは表面上で常に同じピンの密度を使うことができ，導線は面倒をかけない．この種の 3 次元設計での問題は，もちろん，誰かがこれを調べるためには──何が起こっているのかを見るためには──手や道具を中に入れて，その中に入れなければならないことである．少なくとも 2 次元では上から回路を調べることができる！

今までどおり，レント規則の妥当性を仮定して，もう 1 つの興味深い質問をすることができる．「コンピューターの中の導線の長さの分布はどうなのだろうか？」 セルと導線で覆われたボードの，大きな 2 次元のコンピューターがあると仮定する．何本かの導線は短くて，おそらく隣接したセルの間を渡っているかもしれないが，ボードを横切って伸びている導線があるかもしれない．当然，出てくる疑問は，「導線をランダムに選ぶと，その導線がある長さである確率はどうなるだろうか？」である．レントの規則を使うと，一応は，この確率を実際に推測できる．図 7-60 に示されている 2 次元の場合にもどり，今度は L をボード上の任意のユニットの辺の長さとする．このユニットの内部にある 2 つのセルを接続するすべての導線の長さを L 未満であると考えることができる．もちろん，対角線の配線もありうるので，これは厳密には正しくない．しかし，大きさのオーダーだけを扱うのなら，このような細かいことは無視できると仮定する．また，このユニットを出てボード上の他のユニットに接続する導線があるだろう．これらの導線の長さは L より長いとする．レントの規則から

L より長い導線の数を計算できる──これはユニット上の端子の数の T だろう．この場合に T は次式で与えられる．

$$T = t(L/l)^{2r} \tag{7.42}$$

さて，われわれはランダムに選んだ導線が L より長い確率を計算できる．これは式(7.42)の右辺をユニット上のすべての導線の数で割ったものである．この数は $\propto t(L/l)^2$ であることが容易にわかる．そこで，導線の長さが L より長い確率を $P(L)$ と記すと，レンツの式で $r=2/3$ を使えば，明らかに次式が得られる．

$$P(L) \propto (L/l)^{2r-2}, \quad \text{すなわち} \quad P(L) \propto 1/L^{2/3} \tag{7.43}$$

この統計学をもっとつづけられる．標準的な方法で定義された確率密度 $\rho(L)$ を導入する──長さが L と $L+\delta L$ の間の導線を見つける確率は $\rho(L)\delta L$ である．その結果，次式が得られる．

$$P(L) = \int_L^\infty \rho(L')dL' \tag{7.44}$$

ここで

$$\rho(L) = -dP/dL \propto 1/L^{5/3} \tag{7.45}$$

である．特に，興味ある量の，平均の導線の長さを計算しよう．通常の統計学の推論によって，これは

$$\left[\int_l^{L_{\max}} L\rho(L)dL\right] \Big/ \left[\int_l^{L_{\max}} \rho(L)dL\right] \tag{7.46}$$

である．式(7.46)の積分の区間を手直ししたことを注意しなさい．長さ L の範囲をゼロから無限大までにすると，分子は積分が L について正の次元なので，上限(無限大)で困難が生じ，分母は積分が L について負の次元なので，下限(ゼロ)で困難が生じる．したがって，L に対する上限を $L_{\max}$ とし，下限を自然な値のセルの大きさ l とする．君たちは平均の導線の長さを求めるために式(7.46)の積分ができるだろう．答は

$$2l(L_{\max}/l)^{1/3} \tag{7.47}$$

である．この量は発散することに注意すること──機械(その大きさは，おおよそ $L_{\max}$ である)が大きくなるのにつれて，平均の導線の長さは長くなる．これは驚くべきことではない．しかし，式(7.47)を牛耳っているのは**セルの大きさ**であることに注意しなさい．すなわち，平均の長さが機械の大きさに伴って変わる率は，セルの大きさ(われわれの模型ではセルの間隔に相当)に伴って変わる率に比べて半分である．セルの間隔をもう少し広げると，機械の大きさは不つりあいに膨らまねばならない．

80年代の初期には，良い設計者は少しの発明の才と勤勉さによって，レントの規則以上に回路の詰め込みができると言われていた．しかし，最終製品になったときには，何かが常に生じ──余分な回路が必要だとか，抵抗をここに付けなければならないとか，インダクタンスをそこにとかで──，機械が完成したときには，この機械はレントの規則に従っていることがわかったものだった．特定の回路でレント

の規則が破れることはありうるが，最終製品になったときには，レントの規則が支配する．今日では，機械の中身が占める空間を最小にするように中身を配列しようとする半知能的なソフトウェアの「機械の詰め込みプログラム」がある．

編者の最終コメント

言うべきこととして残っているのは，われわれがこの本に入れようとしなかった録音されたファインマンの講義が若干あるということである．これらの講義は光ファイバーの物理や光コンピューターの可能性などの興味深い話題を含んでいる．しかし，これらの分野ではその後の技術的な発展が著しく，このような話題は続巻での専門家による最新の概観にゆだねるのが最善だとわれわれは考えた．この編集者の注意を含めると，この本に含まれている講義は計算分野についてのファインマンの講義を正確に再現している．さらに，彼が話題を選択したこれらの講義は，彼が計算機科学の将来にとって重要だと感じた領域を示している．

あとがき：ファインマンの思い出

私は1970年10月のある陽気な日の朝，カルテクに到着したときのことをよく覚えている．当時のオックスフォードでは大学院の学生でもワイシャツを着てネクタイをしめていたが，そこを卒業したての私としては，マレー・ゲルマンに初めて会いに行く際に何を着ればよいのか見当がつかなかった．一か八かでスーツを着て，理論グループの秘書ジュリー・クルシオのオフィスまで行ったのだが，これが間違いで，まるで「オックスフォードを出たばかりのPh. D.」と書いた大きな札を首からぶら下げているような居心地の悪さを味わった．私は前に一度，イギリスでゲルマンに会ったことがあったが，開襟シャツの個性的な服装でジュリーのオフィスに座っているあごひげの人物があの有名な教授かどうかよくわからなかった．私が自己紹介をしたら，「やあ，私がマレーだ」と言いながらゲルマンが手を差し出してきたので，私の疑念は晴れたのだった．このエピソードは私がカリフォルニアで経験した(健全な)カルチャーショックのほんの一部にすぎない．オックスフォードでの6年間で，私は指導教官を「Professor Dalitz, sir」と呼ぶように慣らされたのだった．そのころは，ダリツをファーストネームで呼ぼうなどとしたことはなかった．

パサデナに着いて最初の仕事の1つは車を買うことだった．これはそれほど容易なことではなかった．パサデナの中古車店はコロラド大通りに沿って数マイルにわたって典型的なアメリカ風に散在しており，ロサンゼルスの公共輸送機関が衰退しておそらく最低の時期だったこともあり，そこにたどり着くのは容易なことではなかった．妻と私が警官に呼び止められ，なぜパサデナの通りを歩いているのかと質問されて初めて，私は，カリフォルニアでは車を買うために車を持っていなければならない，というパラドックスがわかった．別の「鶏と卵」の問題が，以前には出会ったことのなかった「ID」(身分証明書)という言葉に関連して起こった．警官はわれわれにいつもIDを示すことを要求したが，当時のパサデナでは，ただ1つの受理可能なIDは，もちろんカリフォルニアの運転免許証だった．持ち主の写真がないイギリスの運転免許証は明らかに不適当で，パスポートでさえ疑いの目で見られた．中古車のセールスマンたちを通じてのアメリカの初体験は，私の最悪の敵にさえ薦められないものであり，私がカルテクの大学院生に助言を求めたことは驚くにあたらない．私はスティーブ・エ

リスの指示に注意を向けた．彼はデトロイト出身でクルマ社会に詳しいと信じられていたため，そのアドバイスは重んじられていた．私はスティーブをセミナー室まで追いかけたが，そこで彼は，私が最近出会った中古車のセールスマンに似た温和に見える人物と議論していた．もちろん，これがディック・ファインマンとの最初の出会いだった．『ファインマン物理学』の3冊の赤い本に出ている昔の写真しか知らなかったので，すぐには彼とはわからなかった．不思議なことに，10年以上後になっても，私は彼を「ディック」と呼ぶより，「ファインマン」と呼ぶ方がいつも気楽に感じていた．

オックスフォードの大学院生の頃の生活に比べると，カルテクでの生活に適応することは高速道路で高速車線に移るようなものだった．まず最初に，世界の中心にあるオックスフォードの代わりに，第1近似として，ヨーロッパやイギリスがそこに存在しないことは，明らかだった．2番目に，ファインマンやゲルマンの理論グループの気風としては，物理学とは現在未解決の基礎的な大問題だけを攻撃するものであり，難しいが基本的にはよく理解されている領域で先取権を得ようとするのではないことにすぐ気づいた．私は物質のクォーク描像の共同発見者の1人のジョージ・ツバイグに，私の論文へのコメントを求めたときのことを覚えている．この論文は，スタンフォード線形加速器センター(SLAC)の実験家の友人といっしょに書いた，3体の終状態の解析に関するあまり有名でないプレプリント「SLAC-PUB 1000」であった．ジョージの私への彼らしくない穏やかなコメントは，「結局，われわれは回転不変性を理解している」というものだった．事実，この論文は役に立ち，間違っていないが，カルテクの判断基準では，科学の余白にいたずら書きをしているに等しいものだった．その頃，私はツバイグぐらいの良い物理学者でありたいと憧れていた．この抱負は，量子力学草創期のハイゼンベルクやボルンよりも，彼らの共同研究者のヨルダンの業績を見習いたいと望んでいたようなものに感じられる．

カルテクで最も魅力のあったことの1つは，ファインマンとゲルマンのまわりにいることによる気分の昂揚だった．短くて慌ただしい視野の狭い研究しか経験していないイギリスから来たポスドクとして，私は大学院の最終学年の学生とほぼ同年齢だったので，社会生活の多くを彼らとともに過ごした．ファインマンは彼らのうちの2人，つまりフィン・ラブンダルと，博士号を授与されたばかりのマーク・キスリンチャーといっしょにファインマン流のクォーク模型を活発に研究していた．おそらくラブンダルやキスリンチャーとの研究のため，ファインマンは最終学年の大学院生と密接にかかわっており，われわれはほとんど毎日「グリージー(脂で汚れた)」と呼ばれていたカルテクのセルフサービスのカフェテリアで彼といっしょに昼食をとっていた．

言うまでもなく，われわれのテーブルは常に魅惑の中心地だった．頻繁に議論された話題の1つは，SLACでの電子と陽子の散乱で得られた新しい実験結果についてのファインマンの解釈だった．ファインマンの「パートン(部分子)模型」——陽子が点状の構成要素からできているという直観的な描像——は大成功を収めていたが，マレーにとってそれはイライラの種だった．パートン模型を研究し，その発明者であるファインマンからこの模型について直接聞けることを期待して私が意気込んでオックスフォードを出発したのは当然だった．奇妙なことに，ファインマンのパートンに関するただ1つの刊行論文は陽子-陽子散乱への応用だった．ファインマンがSLACを訪れ，実験家たちが彼に電子と陽子についての驚くべき結果を話したとき，彼はこれがパートン模型で簡単に説明できることを認識した．そこですぐに，ファインマンはSLACでセミナーを行ない，彼らの結果をパートンを使って説明した．彼はこれについては何も書いていない．「ファインマンのパートン模型」による実験結果の解析を論文にまとめることは，そのときにはSLACに不在だったブジョルケンと，SLACのポスドクのパスコスに任された．

研究面での私とファインマンとの最初の出会いではおびえさせられた．カルテクの2人の実験家バリー・バリッシュとフランク・スキウリは，ニュートリノ-陽子実験を提案して，それがちょうど受け入れられたところだった．私は実験家と共同研究をしたがっていたので，彼らのグループの非公式の昼食セミナーで彼らの実験への「パートン模型」の適用について説明するように依頼された．実験グループに話をしに行ったとき，聴衆の中にファインマンを見つけたときの私の驚きを想像してほしい．とにかく，私は話を始め，ファインマンから点を取ることさえできた．講義が始まって間もなく，彼はある関係をどのようにして導いたのかという質問をした．今となっては向こう見ずに思われるが，私は「保存ベクトル流理論を使いました．あなたはこれを発明したのだから知っているはずです！」と答えた．実のところ，セミナーのほとんど終わりに近づくまですべてがうまくいった．私がどのような予言をもっとできるかについて概要を述べようとしたとき，ファインマンが次のように言った．「ストップ．そこに線を引きなさい．その線の上のものはすべてがパートン模型で，その線の下はただのブジョルケンとパスコスの推測にすぎない」．ファインマンがこの点に敏感だった理由は，マレーが「パートンは間抜け」とか「パートン模型が予言するものを知りたい人はファインマンのはらわたを調べろ！」と唸り声で言いながら，カルテクのローリッツェン棟の4階を歩いていたためだということに，私はすぐ気づいた．事実，私のセミナーでのファインマンの線の上側にあったすべての結果は，マレーによるもっと洗練された代数的技法を使って導かれた予言と同じだった．ファインマンは他人による粗っぽい「パートン模型」の予言と自分と

は無関係だとして，彼の簡単で直観的なパートンによる手法がゲルマンのもっとハイカラな方法と同一の予言を与えることを強調したかったのである．不幸にも私の講義は，彼がこの主張をするための手ごろな場所になったのだった．

もちろん，ファインマンやゲルマンと同じグループにいることによる不利なことがあった．私はファインマンのパートン模型を研究するという，かたい目的をもってカルテクに来た．私が気づいていなかったのは，カルテクがパートンの研究を出版できない場所だということだった！　なぜできなかったのか？　パートン模型をゲルマンが明らかに嫌っていたことはあったが，「ファインマンのノート」という手に余るものに比べれば，これは問題にならなかった．私はアイディアがあるとファインマンのところに行って，彼の黒板に誇らしげに私の分析結果を書いたものだった．そのたびにファインマンは聞き，コメントし，訂正した．そしてその後で，熱力学，回転対称性等々を引っぱり出し，いろいろな方法を使って，私の「新しい」結果を導出した．彼は多くの異なる物理的手法で同じ結果を導出できれば，その正しさへの確信が増すと私に説明した．これは非常に教育的で刺激的だったが，また少々私を落胆させ，イライラさせるものでもあった．結局，ファインマンがすでに知っていて，彼の有名な研究「ノート」に書かれていて，彼が発表しなかった結果を，印刷物として発表することはできなかった．そのため私は少し絶望的になって，研究の枠組としてもっと形式的なゲルマンの代数的なアプローチに頼った．助教授のジェフ・マンデュラといっしょに，私は電子と陽子の両方のスピンが同じ方向に揃って「偏極」している場合の電子–陽子散乱を研究した．われわれはパートン模型での等価物が曖昧な新しい予言を発見した．おおざっぱに言うと，パートン模型では高エネルギーの電子との衝突ではパートンのスピンの向きは変わらない．われわれの結果はパートンのスピンの向きが衝突で変化する確率に関するもので，パートン模型では通常は無視されていた「スピン–フリップ」の振幅と関連していた．この新しい結果を持って私はファインマンのところに行き，パートンの方法でこの結果を出すよう彼を挑発した．カルテクでその次の学期に行なった講義で，ファインマンがどのようにこの挑発に応じたかは，講義内容が “Photon-Hadron Interactions” という本になったのでそれを読めばわかる．

ファインマンやゲルマンとともに過ごしたカルテクでの生活では決して退屈しなかった．彼らの英雄的行為の物語が豊富にあった――ファインマンについての多くのエピソードは，今では彼の友人のラルフ・レイトンによって後世の人々のために『ご冗談でしょう，ファインマンさん』に残されている．だが，他にも多くの物語がある．ある友人が講義室に入ろうとしたときにゲルマンが授業のためにドアのと

ころまでやってきたときの話である．その友人がドアを開けようとしたが，マレーが「待て！」と言いながら止めた．建物の外は猛烈な嵐で，特に凄まじい稲妻が見えたとき，ゲルマンは言った．「いまだ！」そしてとても印象的な雷鳴の轟きとともに，その講義室に入った．

別の広まっていた話は，ゲルマンといっしょに発見した弱い相互作用のV-A模型の発見に関するファインマンの講演についてだった．講演の後で，聴衆の一人が彼のところに来て，「失礼します，ファインマン教授．共同研究の話をするときには共同研究者の名前を紹介するのが普通ではないでしょうか？」と言った．伝え聞くところでは，ファインマンは「そうですね．しかし共同研究者というなら普通は何かをするものです！」と言ったということである．明らかに，このような話は伝わるうちに誇張されるが，この話については私が知っているファインマンの性格とはとてもかけ離れたものに思われたので彼に聞いてみた．彼は笑ってこう言った「たしかに私がそのようなことをするとは君は信じられないだろう！」 私はノーベル賞を受賞し，グウェネスとの結婚で幸せを見つけた後のファインマンを知っているだけだった．それ以前の，もっと人をイライラさせ，攻撃的な彼の描像はファインマンの伝記からは浮かび出てくるが，私はいまだに確信が持てないでいる．

確かに彼は素早く面白い応答をするのを楽しんでいた．ファインマンのこのような特徴は招待講演者によるセミナーの中でしばしば現れた．ある印象的な出来事は，講演者が黒板に講演の題の「ポメロン・ブートストラップ」と書いて話し始めたときのことだった．ファインマンは「その2つは矛盾している」と叫び，聴衆は爆笑した．講演者にとってかわいそうなことに，彼はあるエネルギー領域では理論的な結果が一応は正しいと導いていたが，それらを別の領域に適用しようとした．これはファインマンが憎んでいた種類の学問的不誠実だった．その講演者は全聴衆からの非難をさばくのにとても不愉快な時間を過ごすことになった．とはいえ，ファインマンは自制することもできた――別のセミナーの際にファインマンは私の方に身を寄せてきて，「もしこの人が常連の訪問者でなければ，論破してやったのに！」と囁いた．

ファインマンが「マヤ象形文字の解読」と題する有名な講義をカルテクのベックマン講堂で行なったのはこの頃のことだった．ファインマンの2番目の妻，メアリー・ルーとのメキシコへの新婚旅行の話と，ドレスデン写本の解読の話は『ご冗談でしょう，ファインマンさん』の中に載っている．講義そのものは典型的なファインマンの大傑作だった．この話はファインマンが新しい問題に取り組む方法を実例で見事に説明している．写本の翻訳を調べるかわりに，彼がその写本を最初に理解した人物だと信じさせた．ファインマンは表の中のマヤの棒や点に取り組んで，棒が5つの点に等しいことを解き，ゼロの記号を

見つけた．棒や点は最初は 20 で繰り上がるが，2 回目には 18 で繰り上がり，360 の周期を示す．数字の 584 という数はある場所では目につき，236, 90, 250，および 8 という期間から構成されていた．もう 1 つの目につく数は 2920，すなわち 584×5 で，そばに 13×2920 までの 2920 の倍数表があった．ここでファインマンは本の最後を先に見るのと同等のことを行なったと言っている．彼は 584 という数に結びつけられるものを見つけようと天文学の図書館を調べ，583.92 日が地球から見た金星の周期であることを見つけた．それから 236, 90, 250，および 8 という数は，金星の異なる位相と関係していることを発見した．写本には 11,959 の周期を持つ別の表もあったがファインマンはこれが月食を予言するのに使われたと解釈した．ファインマンは独自の実際的な類推を使って，そのような「魔法の」数を持つマヤの魅力を，車が 1 万，2 万，3 万マイルなどを走ったのを走行距離計を見て子供っぽく喜ぶことになぞらえた．

ファインマンが言っているように，「マレー・ゲルマンはその後で世界中の全言語の言語学上の関係について素晴らしい講義を 6 回行なうことで対抗した」．この連続講義では，マレーは一抱えの本を持って現れ，共通の起源を持つ「スーパーファミリー」への諸言語の分類について話した．彼はいつも英語とドイツ語との類似に注意を喚起するのが好きで，たとえばジョージ・ツバイグをジョージ・トゥウィッグと呼んで喜んでいた．私は，北方，アフリカ-アジア，インド-太平洋，ニジェール-カルドファン，ナイル-サハラのスーパーファミリーの例についての彼のいくつかの講義ノートをまだ持っている．素粒子が専門の物理学者が比較言語学の講義に出席するのは少し奇妙に思えたが，カルテクでの生活ではいつも好奇心がかき立てられたものだった！　ラルフ・レイトンが編集した逸話の第 2 集『困ります，ファインマンさん』に載っているキャッツキル山での父親との思い出は，いくぶんかはゲルマンの言語や名前に対する熱中に向けられていると私は常に感じていた．その話の中で，ファインマンの父親は「世界中の言語でのあの鳥の名前を知ることができるだろうが，たとえそれを知ったとしても，その鳥について何も知ったことにはならないよ」と言っている．ファインマンは，自分が「あるものの名前を知ることと，あるものを知ることとの相違について非常に早くに知った」のは父親とのこの経験のためだと言っている．

ファインマンのいろいろな思い出はまだ私の記憶に新しい．あるとき昼食時にグリージーにコーヒーを飲みに出かけ，もどってきたところ，ファインマンが私の妻を週末にメキシコの彼の家へ招待したことがわかった──彼の家族といっしょにであった．彼は後から急いで私も招待した．われわれはメキシコの海岸を散歩したり，ファインマンと夜遅くまで物理の話をしたりした．その際にファインマンが私にした忠告は「君は小説を読み過ぎだよ」であった．彼の関心は最初はと

ても狭いところに集中しており，人生の後半になって初めて自分の関心の対象を広げた．おそらくこれは良い忠告に違いないが，ファインマンと付き合った年月の間に，彼を真似ること，つまり委員会や管理運営のような，人生で「重要でない」事柄の無視や，物理学の問題を多くの異なる角度から研究するユニークな能力で彼を真似ることは誰もできないことも学んだ．何年も後にカルテクを訪れたとき，アルタデナの彼の家の庭で，ファインマンは自分のベルトをはずして，スピンと統計の規則についての彼の新しい解釈を説明し始めた．後に彼は，物理学での彼の英雄で，反物質の発見者のポール・ディラックの記念講演の中でこのことについて詳しく書いている．彼がこの規則について初等的な説明をすることができないのを詫びている『ファインマン物理学』の出版から20年くらい後のことだった．そのときに彼は，「このことはおそらく，関係している基本原理をわれわれが完全には理解していないことを意味している」と言っているからである．

何がファインマンの講義をユニークにしているのだろうか？　有名なコーネル大学の物理学者デビッド・マーミンは，物理学でよく理解されていると考えられている問題の思慮深く鋭い分析で著名な人物だが，彼は「私はファインマンが都市の下水システムについて講義するのを聞くためにすべての予定を取り消したい」と感動して言った．1967年に『ロサンゼルス・タイムズ』の科学記者は次のように書いている．「ファインマン博士の講演は本当に稀に見るすばらしいものだ．そのユーモア，ドラマ，サスペンスと興味深さはブロードウェイの舞台演技に匹敵する．とりわけ，彼の講演は明快さに満ちている．物理学が科学の根底にある「メロディー」ならば，ファインマン博士はその最も明快な吟遊詩人である．」　同じ記事で記者のアービング・ベンゲルスドルフは，ファインマンの手法の本質について次のようにまとめている．「重力から量子力学を経て相対論に至るまで，問題がいかに難しくても，その言葉は知的に鋭くわかりやすい．もったいぶった言葉使いや，「手の込んだウソ」や，曖昧にすることは無い．」同じ年の『ニューヨーク・タイムズ』の記事には，ファインマンは「ビリー・ローズが舞台で美しい婦人を扱うように，手振りと抑揚を大々的にではあるが優雅に使っている」と書かれている．

私にとって，ファインマンの講演を聴くことをこのようにユニークな経験にしているのは，ファインマンの言葉の選択によってであった．この『ニューヨーク・タイムズ』の記事はつづけて「彼の講演は簡潔で密度が高くしばしば荒っぽい言い回しで表現されている」と書いている．このような例は無数にある．難しい数学のページの真ん中で，ファインマンは「君たちは2つの新しい状態を調理(＝作成)できる…」というような言葉使いを取り入れたり，「さて，興味深い状況があるとゲルマンとパイスは言った」というような物理学者の会話を想

定して持ち込んで説明を擬人化したりして，文章を意図的に明るくしている．1971 年に物理教育への貢献に対してエルステッドメダルを授与された際の招待講演では，ファインマンはまず「私は教え方については何も知りません」と言って，雰囲気を和らげ，それから彼が研究している問題についての魅力的な説明をしはじめた．「陽子は何からできているのでしょうか？　誰も知りません．それをわれわれは見つけ出そうとしているのです．」 その話の中で，彼は 2 つの陽子を互いに衝突させることを，2 つの時計を互いに衝突させることにたとえた．つまり，衝突で飛び出してきた歯車や，その他のこまごましたものをすべて調べて何が起こっているのかを理解しようとすることにたとえた．このようにして彼は，電子のような単純な点状の粒子を陽子に衝突せさることは，調べたい時計が 1 つしかないためにずっと簡単であることを説明できた．ある夏にイタリアのエリチェでの夏の学校で保存則について質問され，ファインマンはこう答えた．「もし猫がパサデナで消え，同時にエリチェで現れたら，これが猫の大域的保存の例になるでしょう．これは猫の保存のされ方ではありません．猫や電荷やバリオンはもっと連続的な様式で保存されます．」

ファインマンのノーベル賞講演は，科学者を志すすべての人にとって必読とすべきものである．その中で，ファインマンは新しい理論を組み立てるために使った足場を取り除くという通例の慣習をやめている．その代わり，彼は自分の偉大な発見への途中で出会った袋小路や間違っていたアイディアのすべてを語っている．この講演でファインマンは「私は科学的に価値がなく，またこのアイディアの発展の理解にも役立たない出来事を詳しくお話しせねばなりません．みなさんにこの講演をもっと楽しんでいただくという目的のためにだけお話しするのです」と述べ，彼の講演のテクニックの多くを披露している．この講演で，相対論的量子力学のやっかいな難問であった無限大にかかわるディラックの挑戦的文章にファインマンが応える試みを始めたいきさつがわかる．ディラックは彼の有名な本の最後に「本質的に新しい物理学的なアイディアがここでは必要とされるように思われる」と書いている．この問題を解決するための彼の若々しいアイディアについて，ファインマンは次のように述べている．「それはすべての始まりであり，そのアイディアは私にとってあまりにも明白で，しかも洗練されているように思われたので，私はそのアイディアと深い恋に落ちてしまった．女性との恋の場合のように，相手をよく知らないので相手の欠点がわからない場合にのみ，このようなことが起こりえます．欠点は後で明らかになるでしょうが，その恋が強ければ相手に引きとめられたままになるでしょう．そこで私は，いろいろな難点があったにもかかわらず，私は若者の熱情でこの理論に引きつけられつづけました．」 その講演の後の方でファインマンは「あるとき突然，私はとても愚かだったことがわかりました．私が記述し，計算してきたもの

はただの普通の反射光にすぎず，輻射の反作用ではなかったのです」と書いている．この20世紀の最も偉大な科学者の一人のさわやかな正直さは，私にとってのもう一人の英雄ヨハネス・ケプラーを思い出させる．彼は物理学の法則を，数学用語による正確で証明可能な言明として書いた最初の人物である．コペルニクスやニュートンとは違い，ケプラーは，火星の軌道が円ではなく楕円であるという衝撃的な結論に至らざるをえなくなるまでの，思考過程のすべての紆余曲折を書き残した．ケプラーは自分の苦闘を「ああ，私は何と頭の悪い老けた鳥だったことか！」と表現した．

この講演の中の最良の思い出の1つは，スロトニックという名の物理学者と，彼の「ケイスの定理」との出会いに関わっている．この話は，ファインマンが彼の「ファインマン・ダイアグラム」が実際に新しいものであることを認識した瞬間の状況を紹介している．その話の全容は次のとおりである．ニューヨークで開かれたアメリカ物理学会の会議で，スロトニックは電子–中性子結合の2つの異なる形式を比較した研究を発表した．長く複雑な計算の後で，スロトニックは，2つの形式が異なる結果を与えると結論づけた．そこで聴衆の一人だったロバート・オッペンハイマーは立ち上がって，スロトニックの計算はケイスの定理を破っているので間違っていると発言した．かわいそうなスロトニックは，その定理を知らないと言ったので，オッペンハイマーはケイス教授が次の日に彼の結果について発表するので聞きなさいと話した．その晩，ホテルでファインマンは眠れなかったので，自分の発明した新しい方法を使ってスロトニックの計算をやり直すことにした．ファインマンは次のようにつづけている．「次の日の会議で，私はスロトニックに会ってこう言った．「スロトニックさん，私は昨晩，あの計算をしました．私はあなたと同じ答になったかどうかを知りたい．私は2つの結合に対して異なった結果を得ました．しかし，私は自分の方法を確かめるために，あなたの答と比べたいのです．」そうしたら，彼はこう言いました．「あなたが昨晩，あの計算をしたとはどういう意味ですか．私は6カ月もかかったのに！」 そして，2人が答を比較したときに，彼は私の結果を見て，「そこの Q，その変数の Q は何ですか？」と聞いたので，私はこう言いました．「それは電子が異なった方向に向きを変えたことによって移行した運動量です．」 すると彼は，「えー?!　私は Q がゼロに近づいたときの極限値の前方散乱しか計算していないのに！」と言いました．私の形式で Q をゼロと置くのはきわめて簡単で，その結果，私の結果は彼と同じになりました．しかし彼は，運動量移行がゼロの場合を計算するだけでも6カ月かかったのに，私は有限で任意の運動量移行の場合を一晩で計算しました．これは私にとって，ノーベル賞を受賞するくらいの感激的な瞬間でした．なぜならば，このことで，とうとう私はある方法と技術を持ち，そして他の人たちがどのようにしたらよいか

がわからないことについて，自分がどのようにすればよいかを理解していることを確信したからです．この瞬間は，私が価値のあることを成し遂げたと実際に認識した勝利の瞬間でした．」ファインマンが講演で述べなかったのは，彼がケイスの発表の後で立ち上がって「あなたの定理は間違っています．私は昨晩スロトニックの計算をチェックして，彼の結果と同じになりました」とその会議で言ったことである．スロトニックのような計算が6カ月もかかった当時，このことは「ファインマン・ダイアグラム」を有名にする出来事になった．

すべての分野の学生にとって，この他に必読すべきは，ファインマンの「カーゴ・カルト・サイエンス(積み荷信仰式科学)」についての文章である*．これはもともと1974年のカルテクの卒業式での講演であったが，その中でファインマンは科学，疑似科学およびどのようにして自分自身をだまさないようにするかを学ぶことについて論じた．講演の統一テーマは「徹底的な科学的良心」の必要性へのファインマンの熱情的信念，つまり研究の応用可能性について資金提供機関に誤解させないこと，たとえ気に入っている理論を支持しない実験結果であっても出版すること，政府に彼らが聞きたくないアドバイスをすること，結果が曖昧にならないような実験を設計すること等々への熱情的信念である．講演の中で彼は，「こう言うのは申し訳ないが，自分自身をだまさないことをどのように学ぶかは，私が知っているどの講義にも特に含まれていなかった．私たちは，諸君が自発的に学んだと希望している」と言っている．新卒業生たちに，「私が述べたような種類の良心を自由に維持できるところ，組織の中での地位や財政的支援などを維持するためにこの良心を捨てるよう強制されると感じないところで諸君が働ける幸運を祈る」と述べて講演を終えた．尊大に響くことを恐れるが，リチャード・ファインマンにそのような環境を提供したことに対して，世界はカルテクに感謝決議をすべきだと私は考えている．ファインマンは特定のある1つの分野を研究するように制約されたことは決してなかった．本書『ファインマン計算機科学』が生まれたのはこの自由の行使による．

あと2つの「ファインマンについての話」を記して，この思い出を終わるのが適切なように思われる．最初の話はロスアラモスでの金庫破りの時代にもどる．1971年にアーバインで開かれた会議で，ファインマンは会議をしめくくるパネル討論のパネラーになることを引き受けた．彼は，物理学者は「大問題」に答えようとすれば成功すると思うかと質問された．ファインマンの答は，「あなたは，われわれが成功するか，と質問しました．私は昔，同じ質問をされた状況を思い出します．私は金庫を開けようとしていました．誰かが私に「何をしているのですか？　成功しそうですか？」と訊ねました．金庫を開け

*　この文章は『ご冗談でしょう，ファインマンさん』に収録されている．［訳者］

るまでは，成功するかどうかわかりません．しかし，多くのナンバーを試してみると，うまくいかない多くのナンバーはわかります！」

第２の話は，本当に最後のファインマンの話である．ファインマンは昏睡状態で，グウェネスは病床の横にいた．彼女は，彼の手が自分の手を握りたそうに動いているのに気づいた．彼女は医師にそのようなことは可能かと訊ねた．医師は，その運動は無意識のもので何も意味しないと告げた．その時，それまで１日半も昏睡状態だったファインマンは両手を伸ばし，それを振って袖を直し，頭の後で両手を組んだ．これは，昏睡状態でも聞いたり考えたりできることを医師に知らせ，いわゆる「専門家」の言うことを常に信じるべきでないと知らせるファインマン流の方法であった．

最後の言葉としては，ファインマンの伝記を書いたジェームズ・グリックの言葉がふさわしい．グリックは，ファインマンの科学に対する哲学を次のような忘れがたい言葉で要約している．

> 彼は疑うことが最も重要であると信じていた．
> 知る能力への汚点としてではなく，知ることの本質として．

トニー・ヘイ
サウサンプトン
1996年３月

訳者あとがき

ファインマンさんが1984-85, 85-86年度にカルテクで開講した「計算機械の可能性と限界」と題するコースの講義を，録音テープとファインマンさんのノートから復元したのが，本書の原書の *Feynman Lectures on Computation* である.

計算機械の限界とは「計算機で何が計算できて何が計算できないのか？　その理由は？　計算に必要なエネルギーはどこまで減らせるか？　計算機はどこまで小さくできるか？」等々を意味している．このような目標を設定して，計算機の機構をその階層構造の基礎から明快に説明し始め，計算機科学をファインマン流に講義している．計算機や情報などの本質を親しみ深い例に基づいて説明しているので，読者は講義に引き込まれ深い理解が得られるだろう.

進歩の速い計算機科学の分野ではあるが，次に記す2つの理由で，この講義はこれからも長い生命を持つと信じている．まず，計算機科学の基本的なテーマについてファインマンさん独自の視点から含蓄のある見方をしていることである．もう1つは，ファインマンさんが「賢かったが無知だった昔の自分に対して自分の考えを説明している」気分で講義している教室の雰囲気が生き生きと伝わってくることである．随所にでてくる問題にどう取り組むかへのコメントはとくに興味深い.

講義は明快であるとはいっても，原書は日本の一般の読者にとって必ずしも理解しやすいものではない．録音を再生した箇所が多いので，1つ1つの文章がいわゆる文法に従っておらず，文章と文章のつながりがスムーズでないところが多い．講義ではボディーランゲージとイントネーションで補われたと思われる箇所の文章は訳しがたい．また，言い間違えたと思われる箇所があり，説明の不十分な箇所もある．編者が最大限の努力を払ったことは窺えるが，ファインマンさんが亡くなっているために確かめられない箇所を含め，本文，式，図にかなりの間違いが見受けられる.

翻訳にあたっては，原書の本文や図の誤りが明白で，それを確実に訂正できた場合には，その事実を記すことなく，訂正した．このほかに原書の表現の意味が不明で編者に問い合わせたがわからない箇所がある．このようなところは原書の英文(録音された英語？)をなるべく忠実に訳した．なお，式でそのような箇所は(5.37)だけである.

訳者の原が本書の翻訳に取り組んだのは今年の4月だった．しかし，

原書のミスと思われるような箇所や難解な箇所について大学の同僚の中山に相談しているうちに中山も共訳者として参加することになった．私は原書の内容が適切に読者に伝わるよう訳すことを心掛けたが，中山の参加で表現がさらに正確かつ平易になった．松田には主として翻訳を効率よく進めるための支援作業を依頼した．講義を最大限に再現するため，編者が多用している「'—'による文中への説明文の挿入」の多くはそのまま再現した．講義でファインマンさんが強く発音した箇所と推測される，原書のイタリックの箇所は，訳書では太字にした．ただ，簡潔で充実した単語や章句を適切に選んで，読者を知的に興奮させ，愉快な気持ちで講義に引き込む，生き生きとした表現の多くが，翻訳では訳者の力量不足で平凡な表現になってしまったことは残念である．

翻訳に際して，術語のカタカナ表記を中心とする理学系と工学系での表記の違いをどうするかに悩んだ結果，ファインマンさんは基本的には物理学者なので，『理化学辞典』第5版，『文部省物理学用語集』などの訳語を使用することにした．Machineは「計算機」と訳さず，あえて「機械」と訳した．カルテクはキャルテクの方が適切だと思うが，『ファインマン物理学』(岩波書店刊)以来の伝統的な訳を踏襲した．訳者の誤解，無知，ミスなどによる誤りがあるかと思う．そのような点があれば，ご指摘をお願いする．

翻訳に際しては，多くの方々に教えていただいたことを，とくに帝京平成大学の大石和男博士には第7章1〜3節の訳文を読んでいただき，貴重なご意見を頂戴したことを感謝している．

編者のあとがきを読むと，私が1963年に渡米してカルテクのポスドクになった頃が懐しく思い出される．ファインマンさんの思い出を書けばきりがないが，1976年にカルテクを訪問し，たまたま参加したファインマンさんの非公式のセミナーでの経験だけを記したい．セミナーの話は良くわかったが，1箇所だけ微妙な点があった．セミナーのあとでファインマンさんは私の所にきて，「君はあそこでこういう疑問をいだいたろう．あそこはこうなんだよ」と黒板に式を書いて説明してくれた．話をしながら聴衆の表情を見て，聴衆の疑問の中身を正確に見抜く能力にはおそれいった．ファインマンさんは，真の教師であった．

翻訳に苦しんでいた頃のある夜，夢の中でファインマンさんに会った．「あなたの本を訳しているが，わからないところがある．ぜひ教えてほしい」と頼んだら，当然「OK」ということだったので，本を取りに行ってもどってきたら，彼は消えてしまっていた．その場ですぐに聞いていたらと残念である．

1998年12月

訳者を代表して
原　康夫

索　引

た　行

な　行

は　行

ま　行

や　行

ら　行

英数字，記号など

訳 者
原　康夫（はら やすお）　帝京平成大学情報学部教授
中山　健（なかやま けん）　帝京平成大学情報学部講師
松田和典（まつだ かずのり）　鳴門教育大学学校教育学部助教授

ファインマン計算機科学　A. ヘイ R. アレン 編

1999年1月26日　第1刷発行

訳 者　原 康夫・中山 健・松田和典

発行者　大塚信一

発行所　株式会社 岩波書店
〒101-8002 東京都千代田区一ツ橋 2-5-5

電 話　案内 03-5210-4000

印刷・製本 法令印刷　カバー 半七印刷

ISBN4-00-005941-6　Printed in Japan

「ご冗談でしょう、ファインマンさん」[全2冊]	ファインマン 著 大貫昌子 訳	B6判平均 300頁 本体 I 1800円 本体 II 1900円
「困ります、ファインマンさん」	ファインマン 著 大貫昌子 訳	B6判 328頁 本体 2000円
ファインマンさんの愉快な人生[全2冊]	J. グリック 著 大貫昌子 訳	B6判平均 450頁 本体各 2330円
光と物質のふしぎな理論 ——私の量子電磁力学——	ファインマン 著 釜江・大貫 訳	B6判 228頁 本体 2200円
ファインマンさん、力学を語る	D.L. グッドスティーン J.R. グッドスティーン 著 砂川重信 訳	B6判 240頁 本体 2200円
科学は不確かだ！	ファインマン 著 大貫昌子 訳	B6判 176頁 本体 1600円
ファインマン物理学[全5冊]	ファインマン, レイトン, サンズ 共著	B5判 318-494頁 本体 3010－4100円

岩波書店刊

定価は表示価格に消費税が加算されます

1999年1月現在